

DevKit8600

AM3359 Evaluation Kit



User Manual

Version 1.2

10th Jan 2014

Copyright Statement

- DevKit8600 and its related intellectual property are owned by Shenzhen Embest Technology Co., Ltd.
- Shenzhen Embest Technology has the copyright of this document and reserves all rights. No part of the document should not be modified, distributed or duplicated in any approach and form without written permission issued by Embest Technology Co., Ltd.
- The use of Microsoft, MS-DOS, Windows, Windows95, Windows98, Windows2000 and Windows embedded CE 6.0 are authorized by Microsoft.

Revision History:

Version	Date	Description
1.0	01/06/2013	Original Version
1.1	07/11/2013	Added Android source code support
1.2	10/01/2014	Localisation

1 Product Overview

1.1 Introduction

The DevKit8600 evaluation kit is a compact, low-cost yet high-performance evaluation board based on the Texas instruments (TI) AM3359 processor. This microprocessor brings together a 720MHz ARM Cortex-A8 low-power application processor with 176K-Byte On-chip boot ROM, and provides ample peripheral interfaces. The DevKit8600 board offers expanded hardware capabilities including a LAN port, an audio input/output interface, USB OTG, USB HOST, CAN interface, RS485 interface, SPI interface, IIC interface, ADC interface, GPMC interface, JTAG interface, TF slot, serial port NAND, and an interface for an optional TFT LCD or touchscreen.

DevKit8600 board can be used for various embedded applications including:

- Gaming Peripherals
- Home and Industrial Automation
- Consumer Medical Appliances
- Printers
- Smart Toll Systems
- Connected Vending Machines
- Weighing Scales
- Educational Consoles
- Advanced Toys

1.2 Kit Contents

The DevKit8600 evaluation kit is available in several different configurations with the following items:

- ✓ DevKit8600 Evaluation Board
- ✓ 2GB TF Card
- ✓ Serial Cable (DB9-DB9)
- ✓ 12V, 1.25A Power Cable
- ✓ USB Cable (Type A Male to Type Mini-B Male)
- ✓ USB Cable (Type A Female to Type Mini-A Male)
- ✓ Ethernet Crossover Cable
- ✓ WiFi Antenna
- ✓ DVD (Including user manual, PDF schematics, Datasheets, Linux 3.1.0 BSP, Android 2.3 BSP and WinCE 7 BSP)
- ✓ **Optional** 4.3" **or** 7" Touchscreen LCD Display

1.3 Board Interfaces

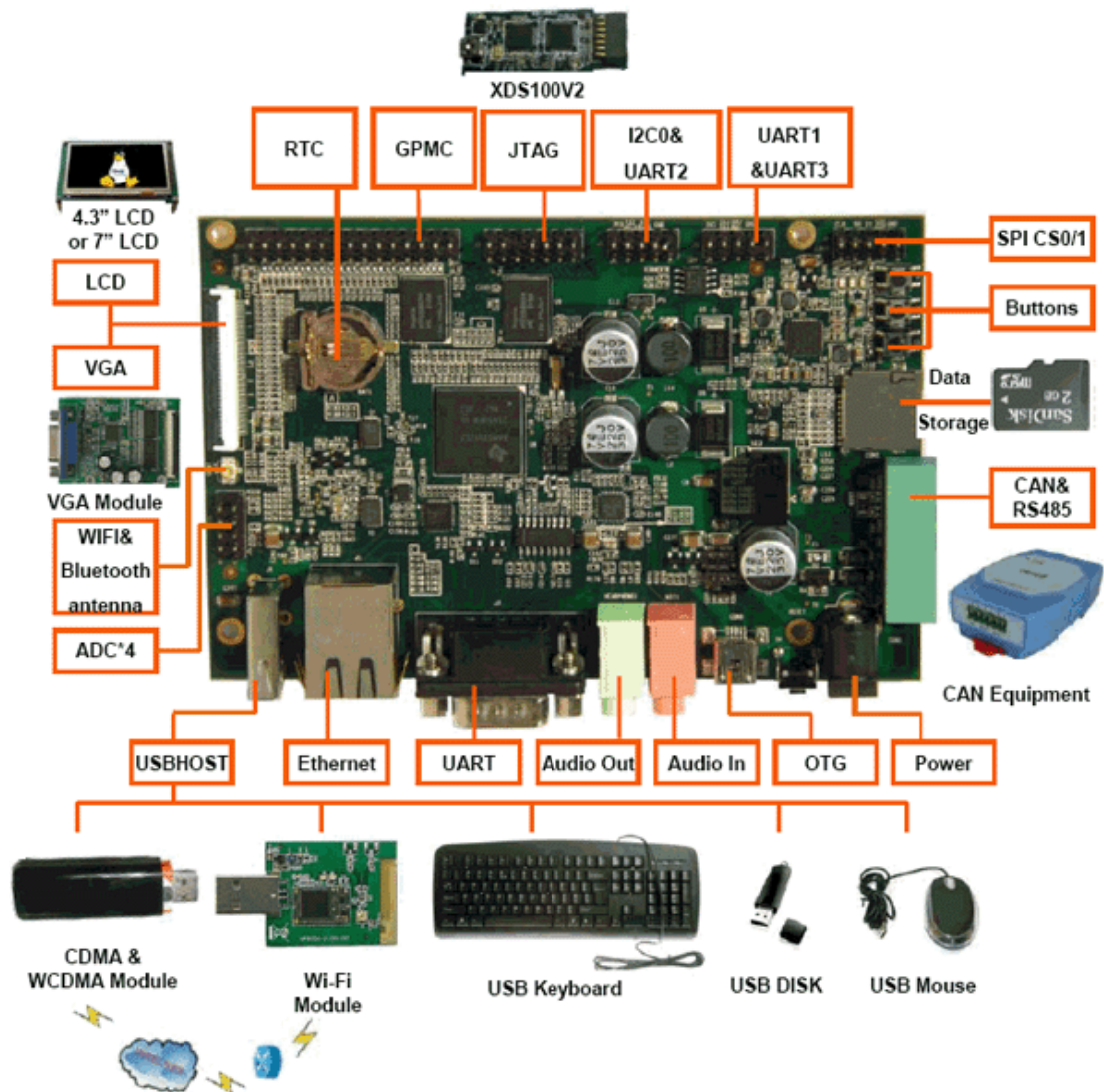


Figure 1: DevKit8600 Interfaces

1.4 System Block Diagram

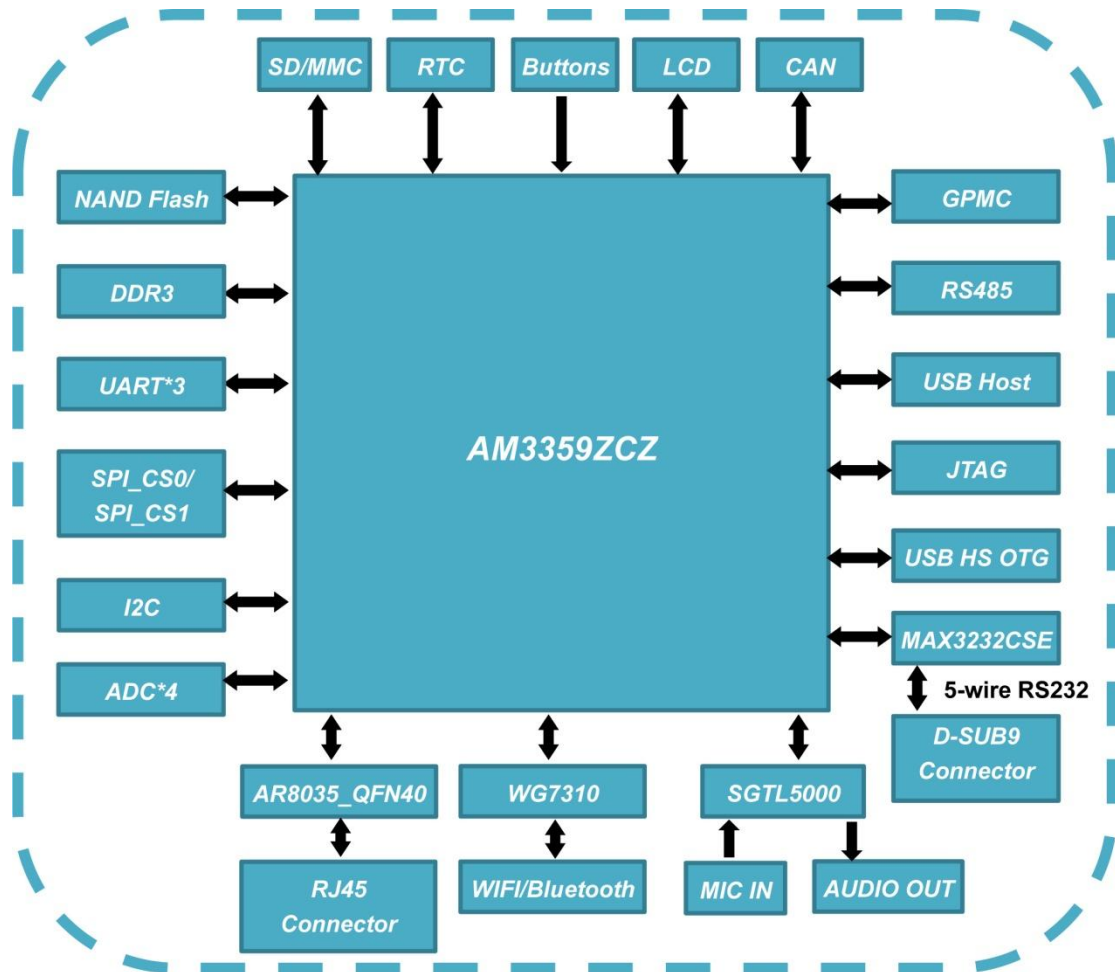


Figure 2: DevKit8600 System Block Diagram

1.5 Physical Dimensions

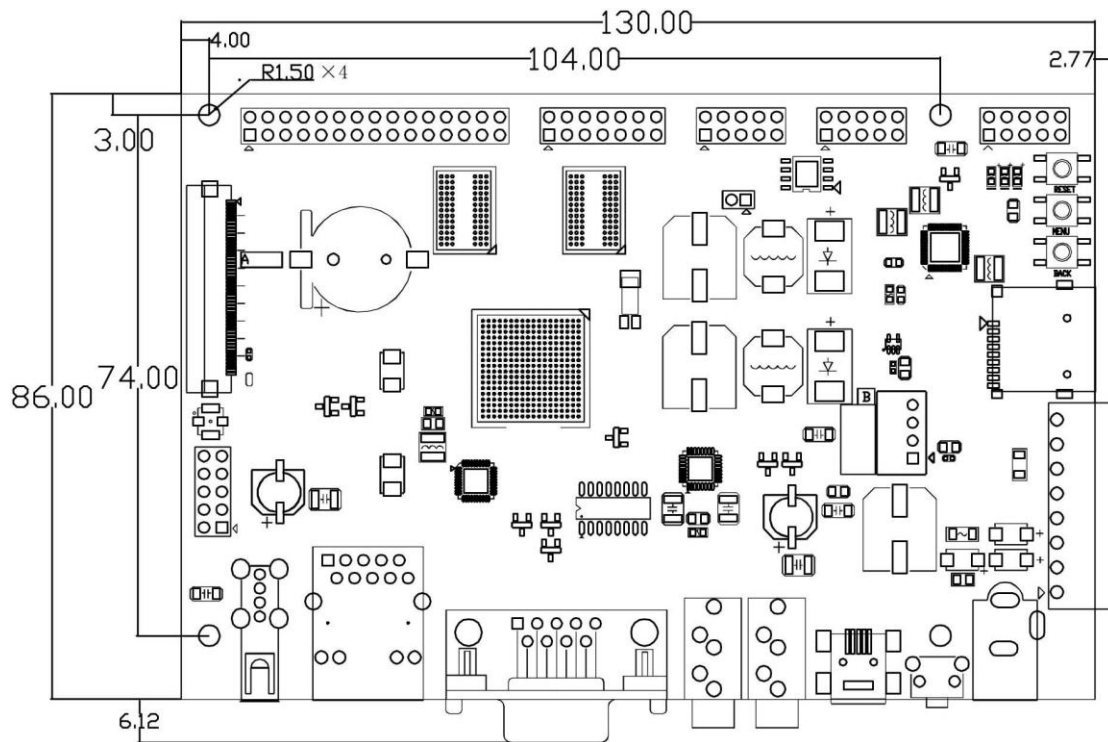


Figure 3: Physical Dimensions

2 Hardware Features

2.1 Processor

- 720-MHz ARM Cortex™-A8 32-Bit RISC Microprocessor
- NEON™ SIMD Coprocessor
- 32KB/32KB of L1 Instruction/Data Cache with Single-Error Detection (parity)
- 256KB of L2 Cache with Error Correcting Code (ECC)
- SGX530 Graphics Engine
- Programmable Real-Time Unit Subsystem

2.2 On-Board Memory

- 512MB NAND Flash
- 512MB DDR3 SDRAM

2.3 On-Board Interfaces

- LCD/Touch Screen interface (Up to 16-Bits Data Output; 8-Bits per Pixel (RGB), 50-pin FPC connector J2)
- Audio input interface (3.5mm audio jack)
- 2-channel audio output interface (3.5mm audio jack)
- 10/100M Ethernet interface (RJ45 connector)
- CAN 2.0 interface
- RS485 interface(8 Pin Phoenix Connector)
- USB 2.0 High-Speed OTG Ports with Integrated PHY (Mini USB)
- USB 2.0 High-Speed HOST Ports with Integrated PHY (USB A)
- Wi-Fi/Bluetooth interface (**only supported by Linux**)
- 5 line Debug serial port, RS232 (DB9 connector)
- Expansion interface
- UART1 interface (J5 connector, compatible with RS485)

- UART2 interface (J6 connector)
- UART3 interface (J5 connector, compatible with WiFi/Bluetooth)
- I2C0 (J6 connector)
- SPI0 interface (J8 connector)
- Four ADC interfaces (J10 connector)
- GPMC bus interface(J14 connector)

2.4 Others

- Three user buttons (HOME, MENU, BACK)
- Reset button
- Power indicator
- Two user configurable LEDs

2.5 Operational Parameters

- Working temperature: 0°C ~ 70°C
- Humidity Range: 20% ~ 90%
- Dimensions: 130x86mm
- Power Consumption: 12V, 0.19A (when booted from Linux without peripherals)

3 Interface Pinouts

3.1 Power Input Interface

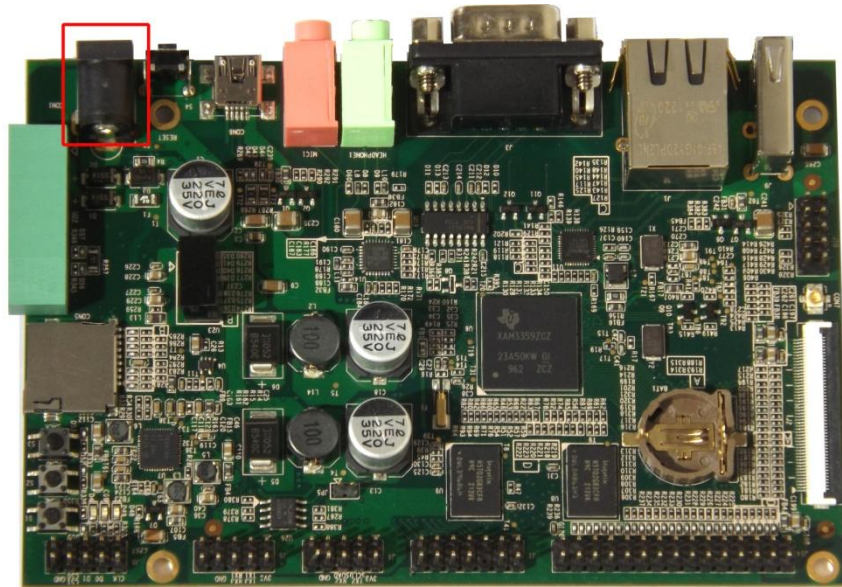


Figure 4: Power Input Location

CON1		
Pin	Signal	Function
1	GND	GND
2	+12V	Power supply (+12V)
3	NC	NC

3.2 TFT_LCD Interface

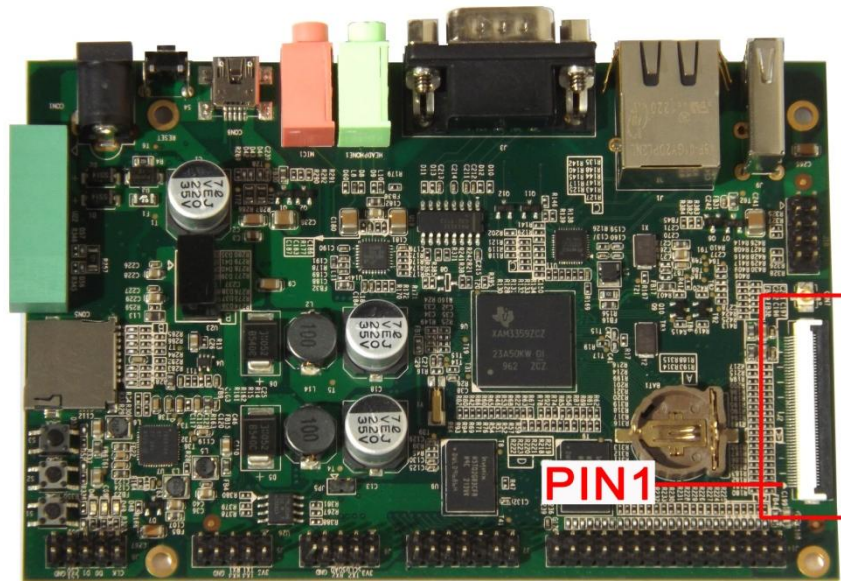


Figure 5: LCD Interface Location



 Do not hot-swap the LCD cable as this will cause damage to the interface!

J2		
Pin	Signal	Function
1	B0	GND
2	B1	GND
3	B2	GND
4	B3	LCD Pixel data bit 0
5	B4	LCD Pixel data bit 1
6	B5	LCD Pixel data bit 2
7	B6	LCD Pixel data bit 3
8	B7	LCD Pixel data bit 4

9	GND	GND
10	G0	GND
11	G1	GND
12	G2	LCD Pixel data bit 5
13	G3	LCD Pixel data bit 6
14	G4	LCD Pixel data bit 7
15	G5	LCD Pixel data bit 8
16	G6	LCD Pixel data bit 9
17	G7	LCD Pixel data bit 10
18	GND1	GND
19	R0	GND
20	R1	GND
21	R2	GND
22	R3	LCD Pixel data bit 11
23	R4	LCD Pixel data bit 12
24	R5	LCD Pixel data bit 13
25	R6	LCD Pixel data bit 14
26	R7	LCD Pixel data bit 15
27	GND	GND
28	DEN	AC bias control (STN) or pixel data enable (TFT)
29	HSYNC	LCD Horizontal Synchronization
30	VSNC	LCD Vertical Synchronization
31	GND	GND
32	CLK	LCD Pixel Clock
33	GND4	GND

34	X+	X+ Position Input
35	X-	X- Position Input
36	Y+	Y+ Position Input
37	Y-	Y- Position Input
38	NC	NC
39	NC	NC
40	NC	NC
41	NC	NC
42	IIC_CLK	IIC master serial clock
43	IIC_DAT	IIC serial bidirectional data
44	GND5	GND
45	VDD1	3.3V
46	VDD2	3.3V
47	VDD3	5V
48	VDD4	5V
49	NC	NC
50	PWREN	Backlight enable

3.3 Audio Output Interface

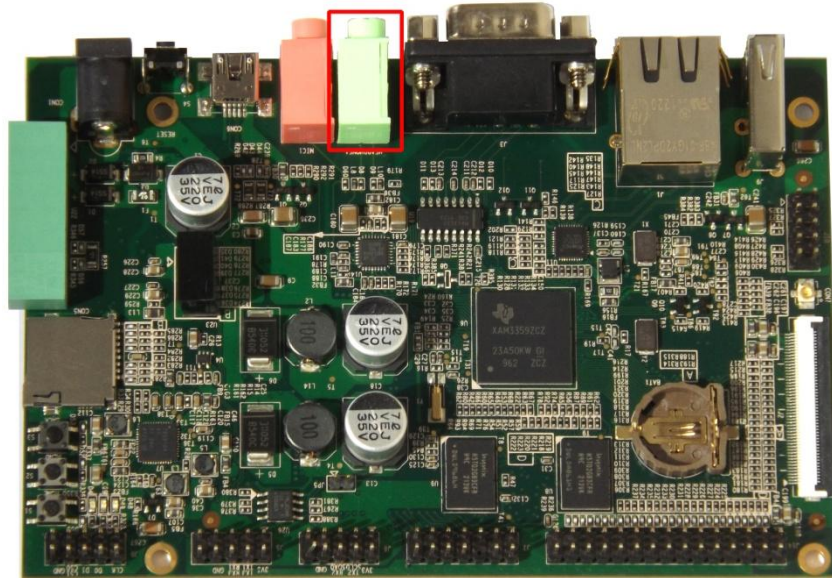


Figure 6: Audio Output Interface Location

Headphone Jack		
Pin	Signal	Function
1	GND	GND
2	NC	NC
3	Right	Right output
4	NC	NC
5	Left	Left output

3.4 MIC In Interface

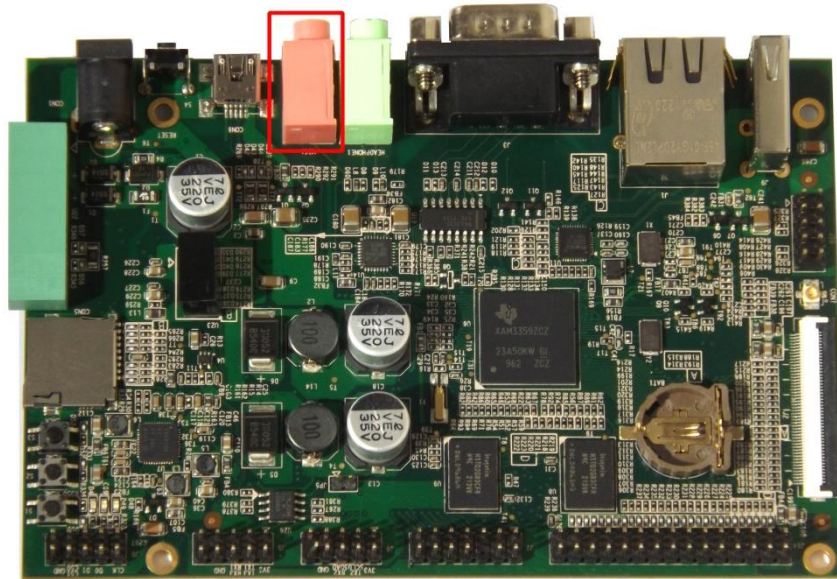


Figure 7: Mic Input Interface Location

MIC1		
Pin	Signal	Function
1	GND	GND
2	NC	NC
3	MIC MAIN P	Right input
4	NC	NC
5	MIC MAIN N	Left input

3.5 USB HOST Interface

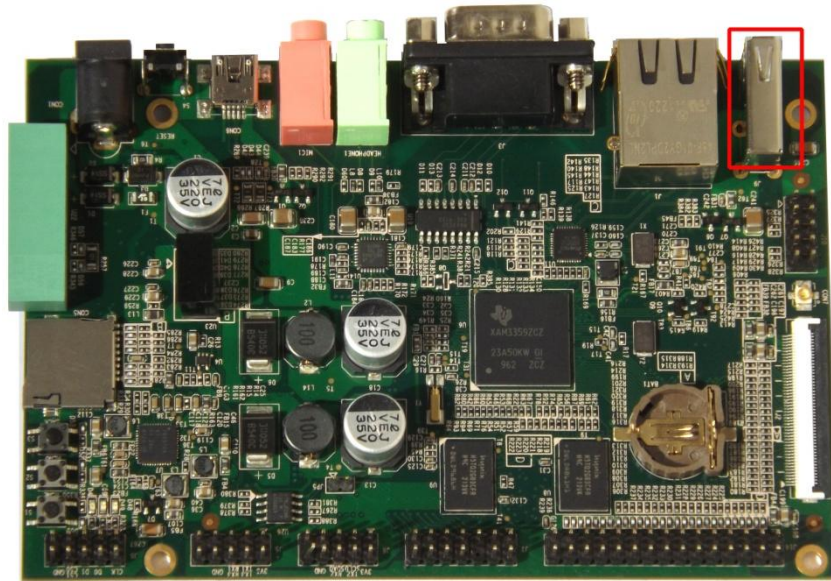


Figure 8: USB HOST Interface Location

J9		
Pin	Signal	Function
1	VB	+5V
2	D-	USB Data-
3	D+	USB Data+
4	GND	GND

3.6 USB OTG Interface

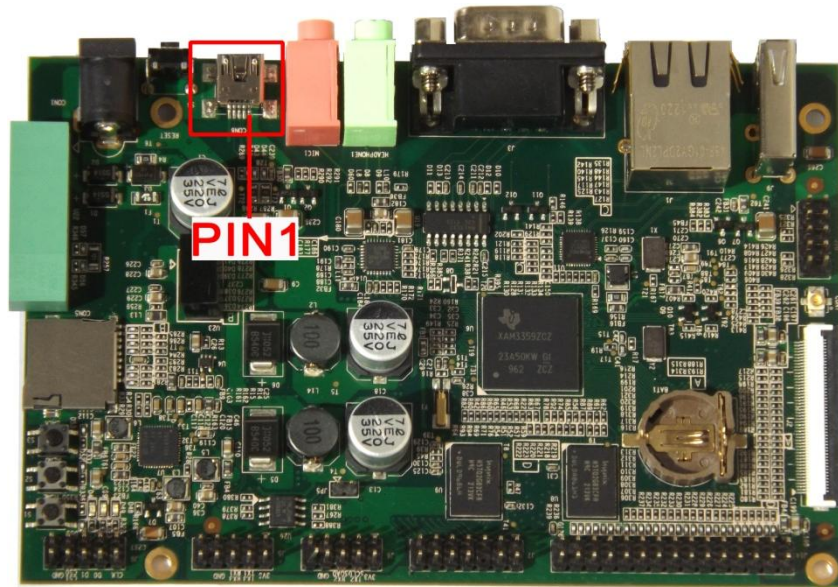


Figure 9: USB OTG Interface Location

CON6		
Pin	Signal	Function
1	VB	+5V
2	D-	USB Data-
3	D+	USB Data+
4	ID	USB ID
5	G1	GND

3.7 TF Card slot

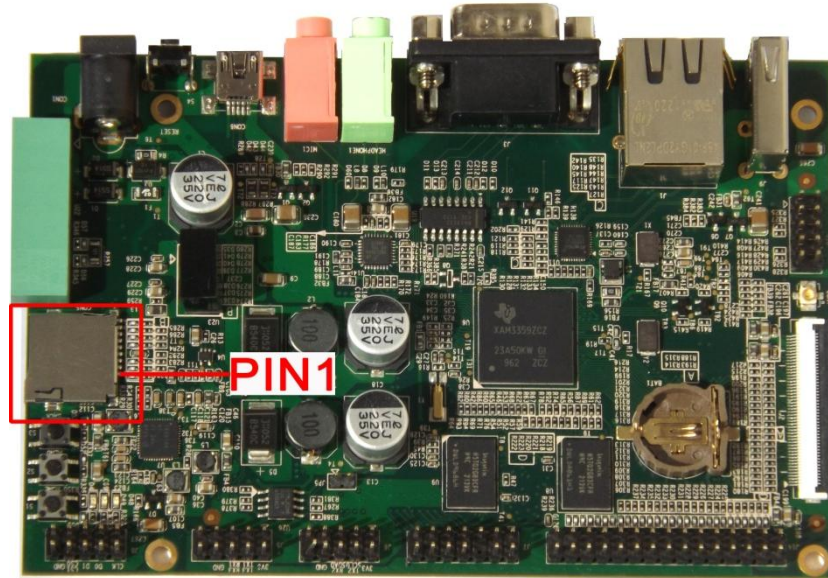


Figure 10: TF Card Slot Location

CON5		
Pin	Signal	Function
1	DAT2	Card data 2
2	CD/DAT3	Card data 3
3	CMD	Command Signal
4	VDD	VDD
5	CLOCK	Clock
6	VSS	VSS
7	DAT0	Card data 0
8	DAT1	Card data 1
9	CD	Card detect

3.8 LAN Interface

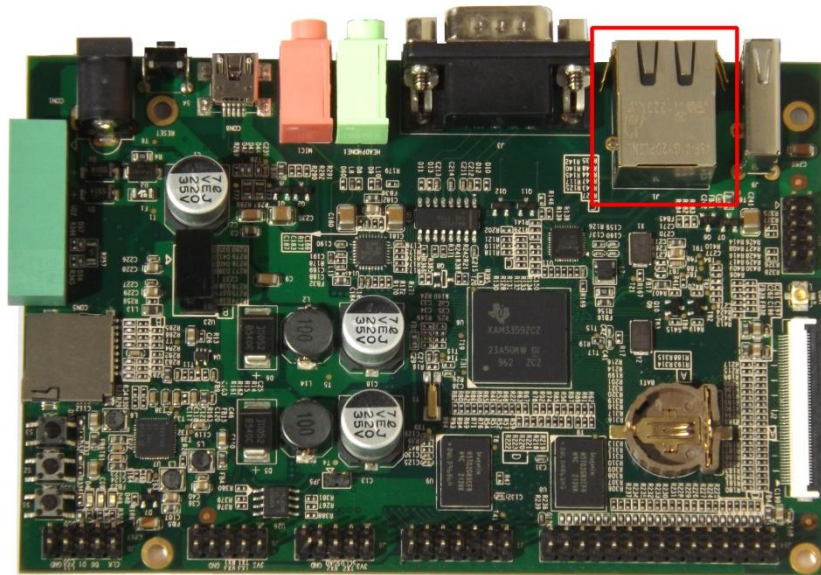


Figure 11: LAN Interface Location

J1		
Pin	Signal	Function
1	TD1+	Transmit Data1+
2	TD1-	Transmit Data1-
3	TDT2+	Media-dependent interface 1, 100 transmission line
4	TDT2-	Media-dependent interface 1, 100 transmission line
5	TCT	Transmit common terminal
6	RCT	Isolating transformer
7	RD1+	Media-dependent interface 2, 100 transmission line
8	RD1-	Media-dependent interface 2, 100 transmission line
9	RD2+	Media-dependent interface 3, 100 transmission line
10	RD2-	Media-dependent interface 3, 100 transmission line
11	GRLA	+2.5V
12	GRLC	LINK active LED

J1		
Pin	Signal	Function
13	YELC	Linked LED
14	YELA	+2.5V

3.9 Serial Port

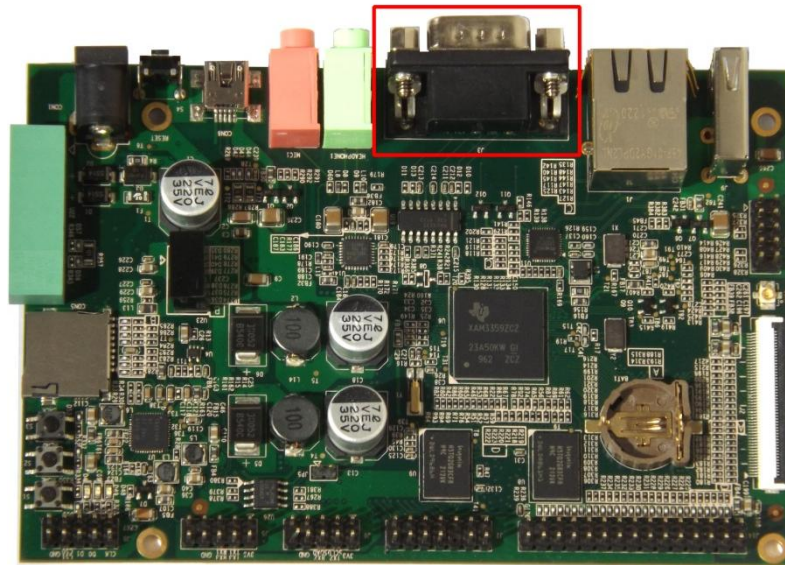


Figure 12: Serial Port Location

J3		
Pin	Signal	Function
1	NC	NC
2	RXD	Receive data
3	TXD	Transmit data
4	NC	NC
5	GND	GND
6	NC	NC
7	RTS	Request To Send
8	CTS	Clear To Send
9	NC	NC

3.10 CAN&RS485 Interface

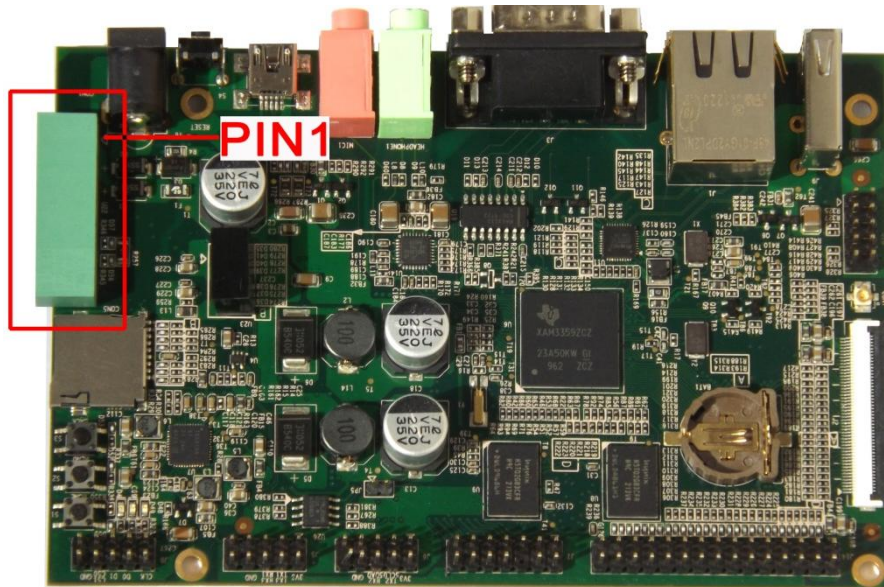


Figure 13: CAN & RS485 Interface Location

U22		
Pin	Signal	Function
1	+12V	+12V
2	GND	GND
3	GND2	Isolated GND
4	485B	485B
5	485A	485A
6	GND1	Isolated GND
7	CANL	CANL
8	CANH	CANH

3.11 JTAG Interface

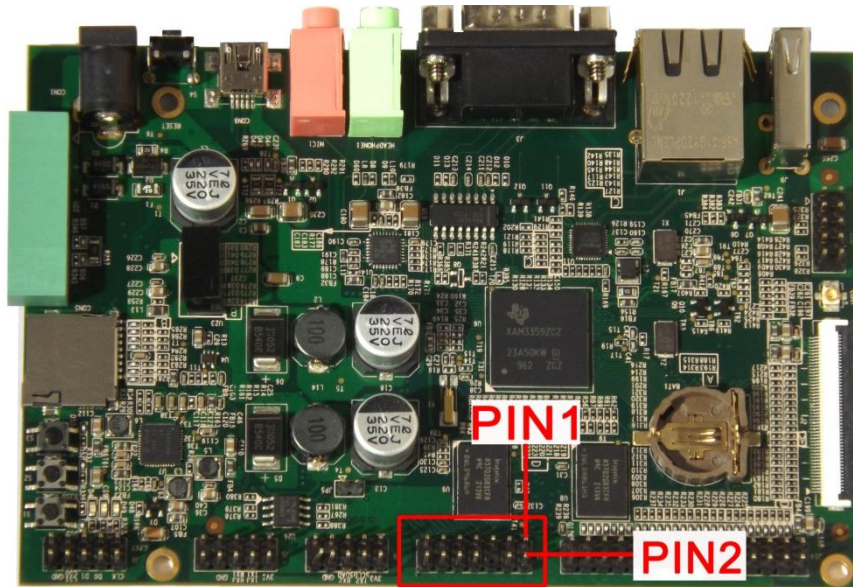


Figure 14: JTAG Interface Location

J7		
Pin	Signal	Function
1	TMS	Test mode select
2	NTRST	Test system reset
3	TDI	Test data input
4	GND	GND
5	VIO	3.3V
6	NC	NC
7	TDO	Test data output
8	GND	GND
9	RTCK	Receive test clock
10	GND	GND
11	TCK	Test clock
12	GND	GND
13	EMU0	Test emulation 0
14	EMU1	Test emulation 1

3.12 ADC

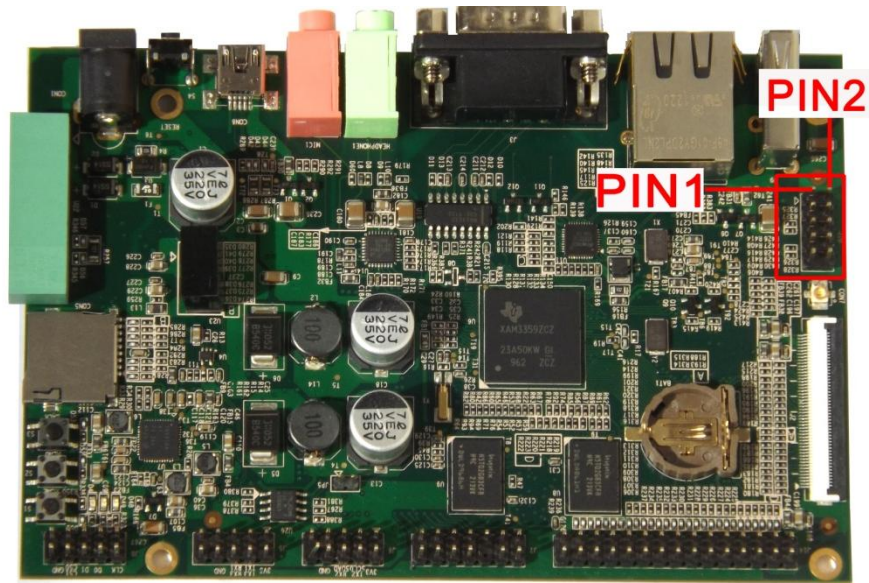


Figure 15: ADC Location

J10		
Pin	Signal	Function
1	GND	GND
2	GND	GND
3	ADC_CH1	ADC1
4	ADC_CH3	ADC3
5	VDDA_ADC	Power
6	VDDA_ADC	Power
7	ADC_CH2	ADC2
8	ADC_CH4	ADC4
9	GND	GND
10	GND	GND

3.13 SPI interface

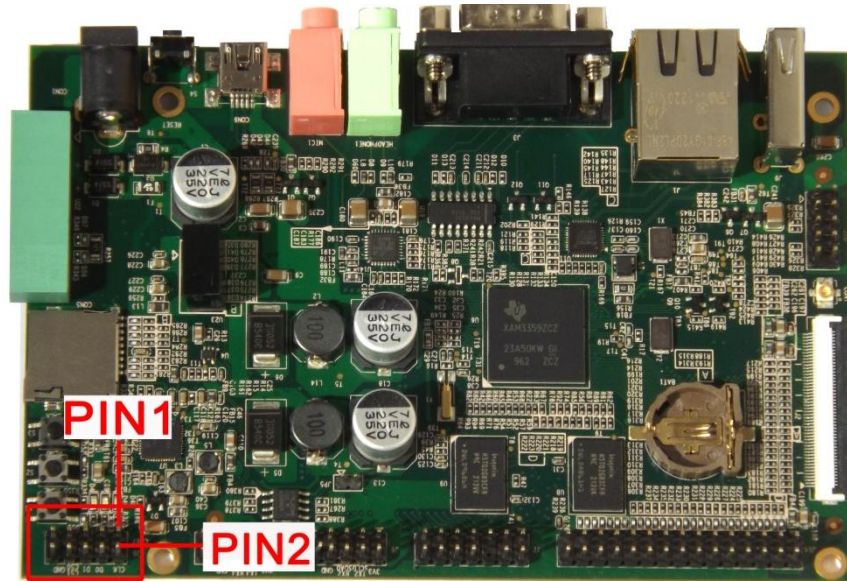


Figure 16: SPI Interface Location

J8		
Pin	Signal	Function
1	SPI_CLK	SPI clock
2	SPI_CLK	SPI clock
3	SPI_D0	SPI DATA0
4	SPI_D0	SPI DATA0
5	SPI_D1	SPI data1
6	SPI_D1	SPI data1
7	SPI_CS0	SPI chip select 0
8	SPI_CS1	SPI chip select 1
9	GND	GND
10	VIO_3V3	+3.3V

3.14 GPMC interface

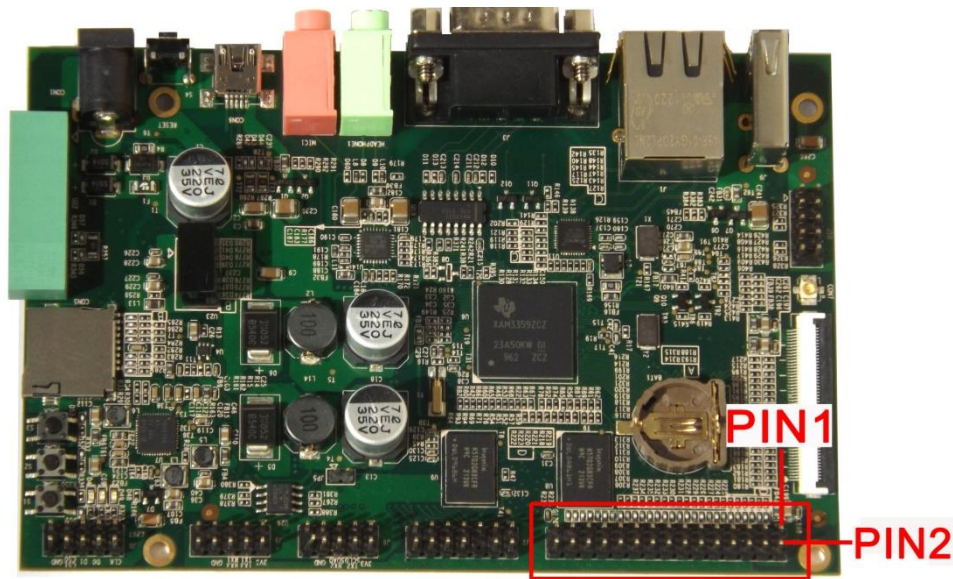


Figure 17: GPMC Interface Location

J14		
Pin	Signal	Function
1	GND	GND
2	VDD3V3_GPMC	3.3V power
3	GPIO0_31	GPIO
4	GPIO0_30	GPIO
5	GPIO1_28_R	GPIO
6	GPIO2_5	GPIO
7	GPIO2_2	GPIO
8	GPIO2_3	GPIO
9	GPIO1_29	GPIO
10	GPIO2_4	GPIO
11	GPMC_A11	GPMC Address
12	GPMC_A10	GPMC Address
13	GPMC_A9	GPMC Address
14	GPMC_A8	GPMC Address
15	GPMC_A7	GPMC Address

J14		
Pin	Signal	Function
16	GPMC_A6	GPMC Address
17	GPMC_A5	GPMC Address
18	GPMC_A4	GPMC Address
19	GPMC_A3	GPMC Address
20	GPMC_A2	GPMC Address
21	GPMC_A1	GPMC Address
22	GPMC_A0	GPMC Address
23	GPMC_AD7	GPMC Address & Data
24	GPMC_AD6	GPMC Address & Data
25	GPMC_AD5	GPMC Address & Data
26	GPMC_AD4	GPMC Address & Data
27	GPMC_AD3	GPMC Address & Data
28	GPMC_AD2	GPMC Address & Data
29	GPMC_AD1	GPMC Address & Data
30	GPMC_AD0	GPMC Address & Data

3.15 Expansion Interfaces

3.15.1 Expansion Interface J5

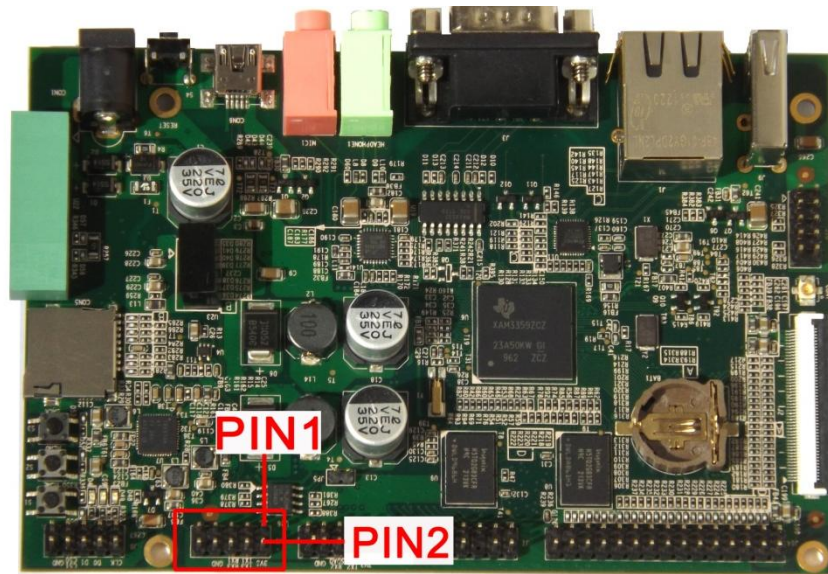


Figure 18: Expansion Interface Location

J5		
Pin	Signal	Function
1	VIO_3V3	+3.3V
2	VIO_3V3	+3.3V
3	UART1_TX_3V3	UART1 Transit data 3.3V level
4	UART3_TX_3V3	UART3 Transit data 3.3V level
5	UART1_RX_3V3	UART1 receive data 3.3V level
6	UART3_RX_3V3	UART3 receive data 3.3V level
7	GND	GND
8	GND	GND
9	GND	GND
10	GND	GND

3.15.2 Expansion Interface J6

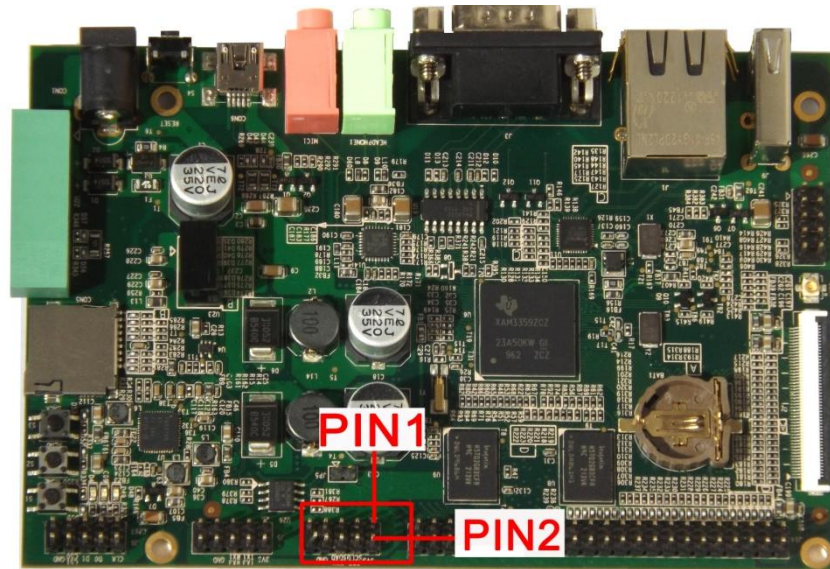


Figure 19: Expansion Interface Location

J6		
Pin	Signal	Function
1	VIO_3V3	+3.3V
2	VIO_3V3	+3.3V
3	I2C0_SCL_3V3	IIC0 master serial clock 3.3V level
4	UART2_TX_3V3	UART2 transit data 3.3V level
5	I2C0_SDA_3V3	I2C0 master serial data 3.3V level
6	UART2_RX_3V3	UART2 receive data 3.3V level
7	GND	GND
8	GND	GND
9	GND	GND
10	GND	GND

3.16 KEY

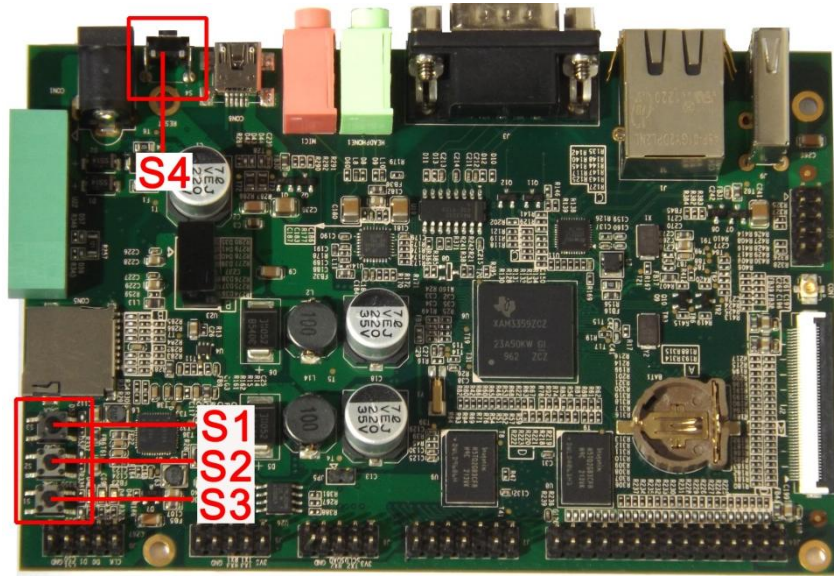


Figure 20: Button Locations

S1-3		
Pin	Signal	Function
S1	HOME	User-defined key
S2	MENU	System menu key
S3	BACK	System back key
S4	SW PUSHBUTTON	Power Switch button

3.17 LED

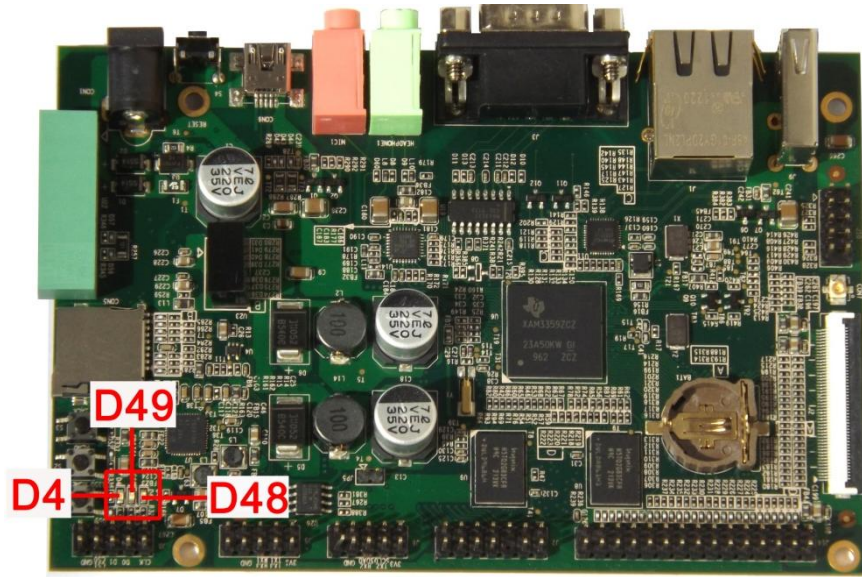


Figure 21: LED Locations

D4,D48,D49		
Pin	Signal	Function
1	D4	System Indicator
2	D48	User-defined LED
3	D49	User-defined LED

4 Software Features

The DevKit8600 Evaluation board is provided with a Windows CE 7.net BSP, a Linux 3.1.0 BSP and an Android 2.3 Demo, with reliable drivers, many of which are in the source code. Please refer to below table for more information.

OS	Item		Content
WinCE7	Bios	X-loader	NAND
		(First Boot Loader)	TF
		EBOOT	NAND
		(Second Boot Loader)	TF
	OAL	OAL module	Boot parameter
			KILT (EMAC)
			Serial debug
			REBOOT
			Watchdog
			RTC
			Kernel profiler
			System timer
			Interrupt controller
			MMU
	Driver	Debug serial port, RTC, Ethernet, NAND Flash, TFT LCD, Touch screen, TF card, USB Device, USB host, Audio input/output, RS485, NLED, Keypad, PRU	
		PowerVR (2D/3D) DDK & SDK	
	APP	Application module	Flash Player plug-in and Flash player

OS	Item		Content
Android Demo (no source code provided)	Kernel	Linux-3.1.0	Gingerbread
	Driver	Debug serial port, RTC, Ethernet, NAND Flash, Touch screen, TF card, USB Device, USB host, Audio input/output, LED, Keypad, 2D/3D	
Linux	BIOS	SPL (First Boot Loader)	NAND
			MMC/SD
			FAT
		u-boot (Second Boot Loader)	NAND
			MMC/SD
			FAT
			NET
	Kernel	Linux-3.1.0	Supports ROM, CRAM, EXT2, EXT3, FAT, NTFS, JFFS2, UBIFS file systems
	Driver	Debug serial port, RTC, Ethernet, NAND Flash, TFT LCD, Touch screen, TF card, USB Device, USB host, Audio input/output, LED, Keypad, CAN, RS485, WiFi/Bluetooth	

5 Linux Operating System

5.1 Introduction

This section is intended to provide detailed instruction on Operating System Software development for the DevKit8600 board. The following topics are covered:

- The Software Resources provided by the DevKit8600.
- The software features.
- The software Development process including how to set up the development environment, building guidance for the boot loader, kernel and file system, and the development of device drivers.
- Flashing methods using boot loader commands.
- The usage of DevKit8600
- Application development.

Prior to this section it is suggested to:

-  Install and configure an Ubuntu Linux environment; please refer to Appendix 1: Installing an Ubuntu Linux System for details.
-  Become familiar with embedded Linux development technology.

5.1.1 Software Resources

This chapter provides an overview of the software related system components of the DevKit8600. A basic software system consists of four parts: spl, u-boot, kernel and rootfs. Figure 22 shows the structure of the system:

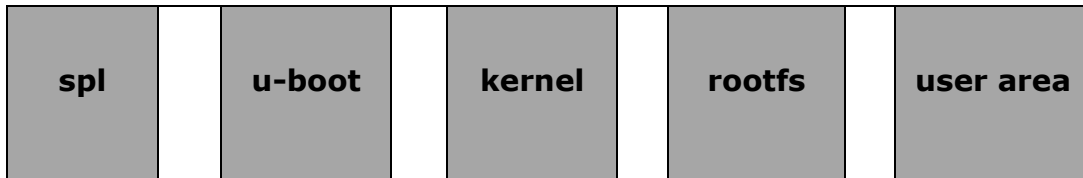


Figure 22: Software System Structure

The features and functions of each part of the system are given below:

1. spl is a first level bootstrap program. After system start-up, the ROM inside the CPU will copy the spl to internal RAM and perform its routine work. Its main function is to initialize the CPU, copy u-boot into the memory and subsequently give control to u-boot
2. u-boot is a second level bootstrap program. It is used for interacting with users, updating images and leading the kernel
3. The latest Linux3.1.0 kernel is employed here and it can be customized based on the requirements of the DevKit8600

rootfs employs an Open-source system. It is small and powerful making it very suitable for embedded systems

5.1.2 Software Features

Items		Notes
BIOS	spl	NAND
		MMC/SD
		FAT
	u-boot	NAND
		MMC/SD
		FAT
		NET
Kernel	Linux-3.1.0	Supports ROM/CRAM/EXT2/EXT3/FAT/NFS/JFFS2/UBIFS and various file systems
Device Driver	serial	Series driver
	rtc	Hardware clock driver
	net	10/100M Ethernet driver
	can	CAN bus driver
	flash	NAND flash driver (supports NAND boot)
	LCD	TFT LCD driver
	Touch screen	Touch screen controller driver
	MMC/SD	MMC/SD controller driver
	USB OTG	USB OTG 2.0 driver
	Audio	Audio driver
	Keypad	GPIO keyboard driver
	LED	User led driver
Demo	Android	Android 2.3.4 system
	TISDK	TISDK system

5.2 System Development

5.2.1 Establishing Operating System Development Environment

Before beginning software development on the DevKit8600, the user first has to install a Linux cross development environment in their computer. The process for this will be introduced below, using the Ubuntu operating system as an example.

5.2.2 Installation of Cross Compilation Tools

Installation of cross compilation tools is done by using the software CD provided along with this kit. To start the process insert the CD and allow it to auto run; Ubuntu will mount the disc under the directory /media/cdrom, the cross compilation tools are saved under the directory /media/cdrom/linux/tools.

The following instructions are executed in the Ubuntu terminal to decompress the cross compilation tools under the directory \$HOME:

```
mkdir $HOME/tools
cd /media/cdrom/linux/tools
tar xvf arm-2009q1-203-arm-none-linux-gnueabi-i686-pc-linux-gnu.tar.bz2 -C
$HOME/tools
tar xvf arm-eabi-4.4.0.tar.bz2 -C $HOME/tools
```

Some of the other development tools used for source code compilation are present in the directory linux/tools of the disc; the user can execute the following commands to copy them to the local folder:

```
cp /media/cdrom/linux/tools/mkimage $HOME/tools
cp /media/cdrom/linux/tools/mkfs.ubifs $HOME/tools
cp /media/cdrom/linux/tools/ubinize $HOME/tools
cp /media/cdrom/linux/tools/ubinize.cfg $HOME/tools
```

5.2.3 Addition of Environment Variables

After all the above tools are installed, it is necessary to use the following commands to add them to the temporary environment variables:

```
export PATH=$HOME/tools/arm-2009q1/bin:$HOME/tools/arm-eabi-4.4.0/bin:
$HOME/tools:$PATH
```

Note:

 The user can add this to the .bashrc file under the user directory, such that the addition of environment variables will be finished automatically when the system is booted. The command `echo $PATH` can be used to check the path.

5.2.4 Building an Android Development Environment

In addition to the installation of cross compilation tools and environment variables, there are some software packages and configurations that need to be handled before you can begin compilation of Android source code. For detailed information, please refer to "Setting up a Linux build environment" on the Android web site:



<http://source.android.com/source/initializing.html>.

5.3 System Compilation

5.3.1 Preparation

Source codes for all the components of the system are under the directory `linux/source` on the disc. The user has to decompress them to the Ubuntu system before starting development:

```
mkdir $HOME/work
cd $HOME/work
tar xvf /media/cdrom/linux/source/u-boot-2011.09-psp04.06.00.03.tar.bz2
tar xvf /media/cdrom/linux/source/linux-3.1.0-psp04.06.00.03.sdk.tar.bz2
tar xvf /media/cdrom/linux/demo/android/source/linux-3.1.0-android.tar.bz2
sudo tar xvf /media/cdrom/linux/source/rootfs.tar.bz2
tar xvf
/media/cdrom/linux/demo/android/source/rowboat-android-gingerbread-am335xevm
.tar.bz2
```

After the above commands are executed, the directories `u-boot-2011.09-psp04.06.00.03`, `linux-3.1.0-psp04.06.00.03.sdk`, `linux-3.1.0-android`, `rowboat-android-gingerbread-am335xevm` and `rootfs` will be created in the current directory.

Note:

 Do not uncompress the source file to any other directory, or errors might occur during compilation.

5.3.2 Bootstrap Program Generation

The DevKit8600 supports TF Card boot or NAND boot. The system will initially attempt to boot from MMC or SD. If this fails it will then attempt to start from NAND Flash.

Below is a sample bootstrap program.

```
cd u-boot-2011.09-psp04.06.00.03
make distclean
make devkit8600_config
make
```

When the above code is executed, the current directory will generate the files MLO and u-boot.img.

5.3.3 Kernel Compilation

Before kernel compilation, the user has to select the correct display mode according to their display device.

For a Linux system, please enter following commands in the terminal window of Ubuntu:

```
cd linux-3.1.0-psp04.06.00.03.sdk
make distclean
make devkit8600_defconfig
make menuconfig
```

For an Android system, please enter following commands in the terminal window of Ubuntu:

```
cd linux-3.1.0-android
make distclean
make devkit8600_android_defconfig
make menuconfig
```

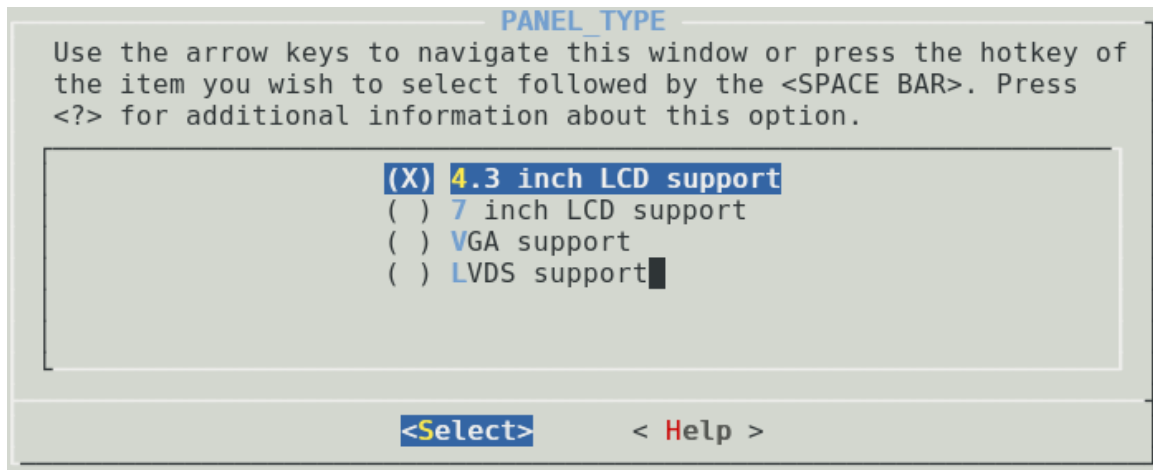
Note:

 If an error occurs in the system when make menuconfig is input, it is necessary to install the ncurses library in the Ubuntu system; the ncurses library is a character graphic library, used in the creation of the menuconfig function of the kernel; the specific installation instruction is: `sudo apt-get install ncurses-dev`

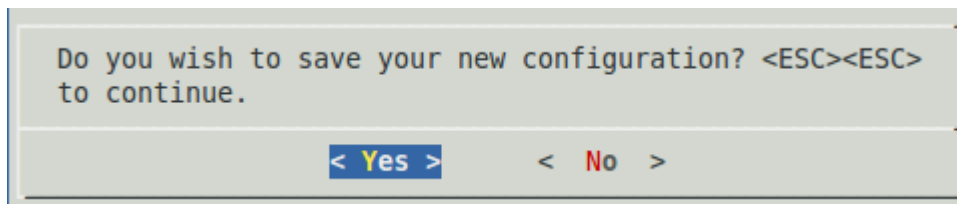
Enter the kernel customization menu; now enter "`PANEL_TYPE`" according to the following pointing paths:

```
Location:
-> Device Drivers
-> Graphics support
-> Support for frame buffer devices (FB [=y])
-> DA8xx/OMAP-L1xx Framebuffer support (FB_DA8XX [=y])
Selected by: HAS_IOMEM [=y] && FB_DA8XX [=y] && m
```

Select the correct screen size under "PANEL_TYPE":



After determining the screen size, return to the parent directory. Select **Exit** to exit, until the following dialog box appears, then select **Yes**:



```
make uImage
```

After above operations are executed, the required uImage file will be generated in the directory arch/arm/boot.

5.3.4 Generation of the File System

1. RAMdisk file creation

RAMdisk creation instructions can be found at:



http://www.elinux.org/DevKit8600_FAQ

2. UBI file making

```
cd $HOME/work
sudo $HOME/tools/mkfs.ubifs -r rootfs -m 2048 -e 126976 -c 812 -o ubifs.img
sudo $HOME/tools/ubinize -o ubi.img -m 2048 -p 128KiB -s 512 -O 2048
$HOME/tools/ubinize.cfg
```

After the above operations have completed, the required ubi.img file will be generated under the current directory.

5.3.5 Building an Android File System

1. Please enter the following commands to compile the source file of the Android system:

```
cd rowboat-android-gingerbread-am335xevm
make TARGET_PRODUCT=am335xevm clean
make TARGET_PRODUCT=am335xevm OMAPES=4.x
```

2. Enter the following command to modify the file Rules.make under hardware/ti/sgx/:

```
Vi hardware/ti/sgx/Rules.make
```

3. Change `KERNEL_INSTALL_DIR=$(HOME)/work/linux-3.1.0-android` to `KERNEL_INSTALL_DIR=/home/user_name/work/linux-3.1.0-android` The `/home/user_name` is the value of `$(HOME)`. You can enter "whoami" in the terminal window of Linux to view the value.

4. Enter the following command to create an ubi file system:

```
source ./build_ubi.sh
```

The ubi.img can be found under temp/.

Note:

 Before you start to compile the Android file system, you need to first compile the kernel source code of Android (linux-3.1.0-android), or errors may occur.

5.4 System Customization

As the Linux kernel has many kernel configuration options, the user can increase or reduce the drivers or some kernel features from the default configuration to better meet their demands. The general process of system customization will be described with examples below.

5.4.1 Modification of Kernel Configuration

A default configuration file is provided in the factory kernel source codes:

linux-3.1.0-psp04.06.00.03.sdk/arch/arm/configs/devkit8600_defconfig

A user can carry out system customization on this basis:

```
cd linux-3.1.0-psp04.06.00.03.sdk
cp arch/arm/configs/devkit8600_defconfig .config
make menuconfig
```

The system customization will be described below using a USB gadget and a USB mass storage device as an example:

Select the configuration below:

-> Device Drivers

-> USB support

-> USB Gadget Support

-> USB Gadget Drivers

```
--- USB Gadget Support
[ ] Debugging messages (DEVELOPMENT)
[ ] Debugging information files (DEVELOPMENT)
[ ] Debugging information files in debugfs (DEVELOPMENT)
(2) Maximum VBUS Power usage (2-500 mA)
(2) Number of storage pipeline buffers
<*> USB Peripheral Controller (Inventra HSDC USB Peripheral (TI, ADI, ...)) --->
<*> Select one gadget as builtin for one port
    Select USB port to bind builtin gadget (USB-0) --->
<M> USB Gadget Drivers
<M>   Gadget Zero (DEVELOPMENT)
< >   Audio Gadget (EXPERIMENTAL)
<M>   Ethernet Gadget (with CDC Ethernet support)
[*]    RNDIS support
[ ]    Ethernet Emulation Model (EEM) support
< >    Network Control Model (NCM) support
< >    Gadget Filesystem (EXPERIMENTAL)
< >    Function Filesystem (EXPERIMENTAL)
<M>    File-backed Storage Gadget (DEPRECATED)
[*]    File-backed Storage Gadget testing version
< >    Mass Storage Gadget
< >    Serial Gadget (with CDC ACM and CDC OBEX support)
< >    MIDI Gadget (EXPERIMENTAL)
< >    Printer Gadget
< >    CDC Composite Device (Ethernet and ACM)
< >    Multifunction Composite Gadget (EXPERIMENTAL)
< >    HID Gadget
< >    USB Webcam Gadget
```

Select "`File-backed Storage Gadget`" as `<M>`, **exit**, and finally select **Save** to recompile the kernel.

5.4.2 Compilation

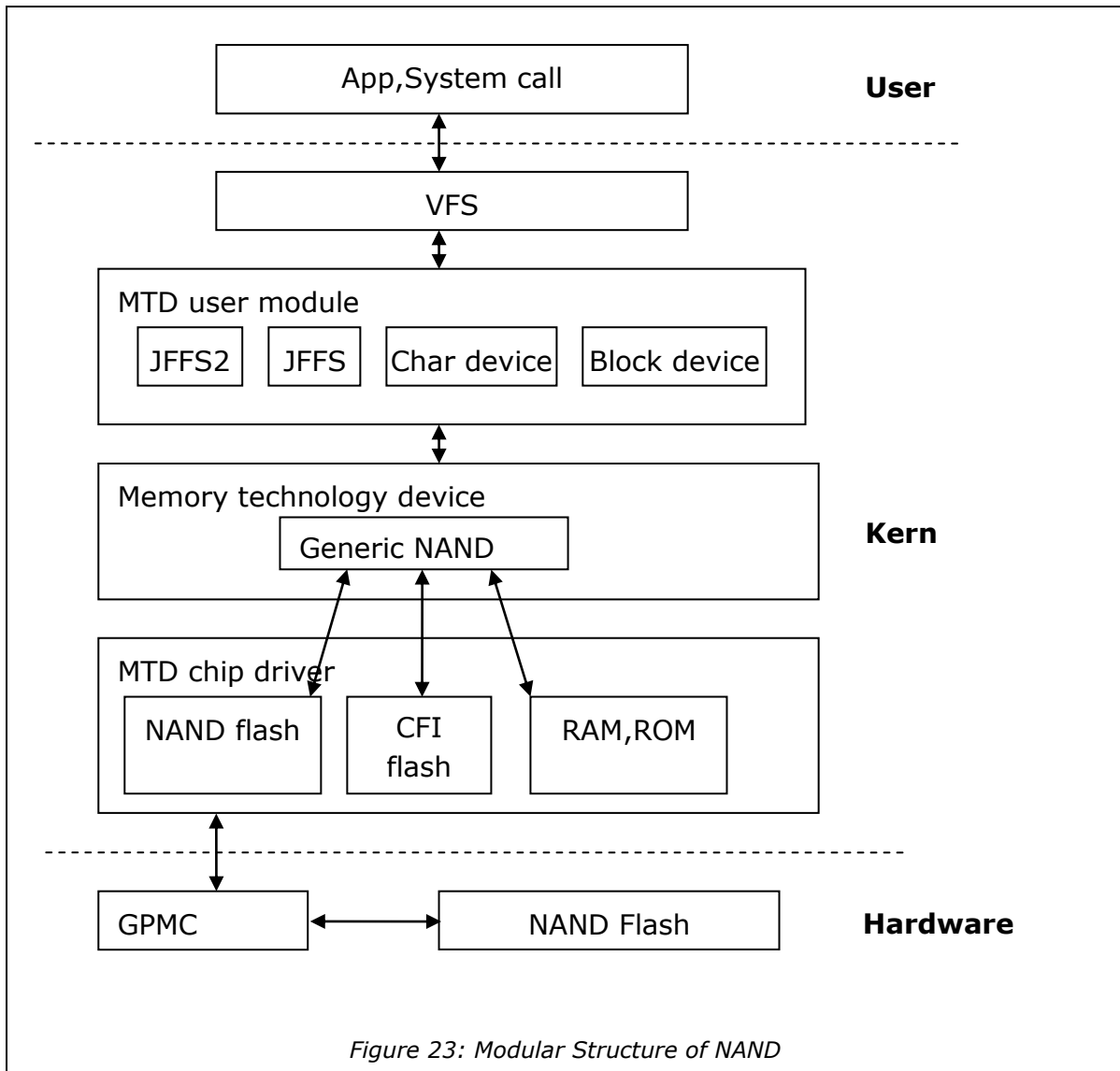
Save the configuration and run the following commands to recompile the kernel:

```
make uImage  
make modules
```

After the above operations are executed, a new kernel image, `uImage`, will be generated under the directory `arch/arm/boot`, and a module file `g_file_storage.ko` will be generated under the directory `drivers/usb/gadget`.

5.5 Introduction to drivers

5.5.1 NAND



The solid-state memory used in embedded systems is mainly flash; this system in particular uses NAND Flash.

NAND Flash is used as a block device, on which the file system is arranged; interaction between user and NAND Flash is mainly facilitated by a specific file system. In order to compensate for different flash memories, the kernel inserts an MTD subsystem between the file system and the specific flash driver for management.

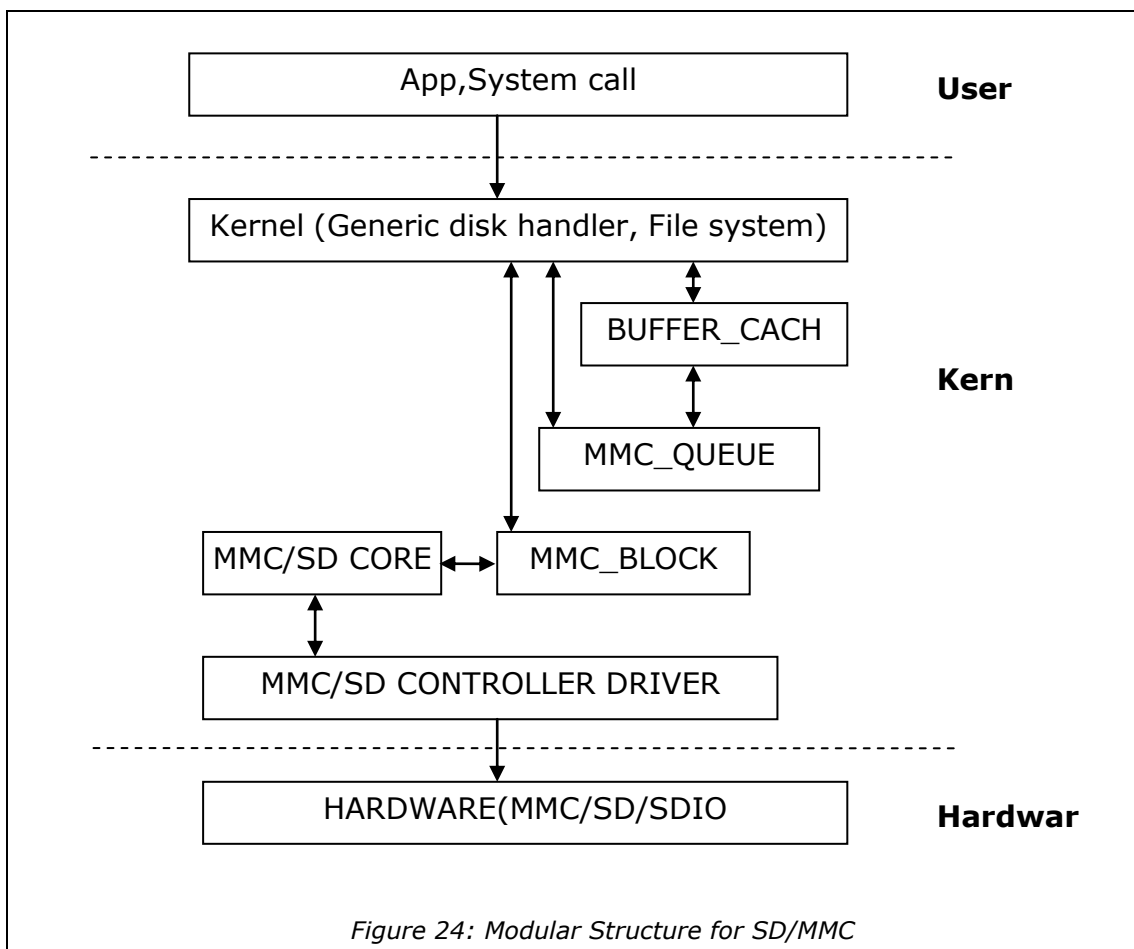
Therefore, the user accesses NAND Flash through the following process:

- 🖱 User
- 🖱 System Call
- 🖱 VFS
- 🖱 Block Device Driver
- 🖱 MTD
- 🖱 NAND Flash Driver
- 🖱 NAND Flash.

Kernel Driver reference paths:

- 📖 linux-3.1.0-psp04.06.00.03.sdk/drivers/mtd/nand/
- 📖 linux-3.1.0-psp04.06.00.03.sdk/drivers/mtd/nand/omap2.c

5.5.2 SD/MMC



The SD/MMC card drivers under Linux mainly include SD/MMC core, mmc_block, mmc_queue and SD/MMC driver:

- **SD/MMC core:** controls core codes unrelated to structure in the SD/MMC card operation.
- **mmc_block:** controls driver structure when SD/MMC card is used as a block device.
- **mmc_queue:** manages the request queue.
- **SD/MMC driver:** manages specific controller drivers.

Kernel Driver reference paths:

 linux-3.1.0-psp04.06.00.03.sdk/drivers/mmc/

 linux-3.1.0-psp04.06.00.03.sdk/drivers/mmc/host/omap_hsmmc.c

5.5.3 LCDC

The LCD controller (LCDC) on AM335x is an updated version of LCDC that is found on OMAP-L138 SoC. It has following upgrades in comparison with OMAP-L138

- Interrupt configuration and status registers are different.
- Increased resolution of 2048x2048.
- 24 bits per pixel active TFT raster configuration.

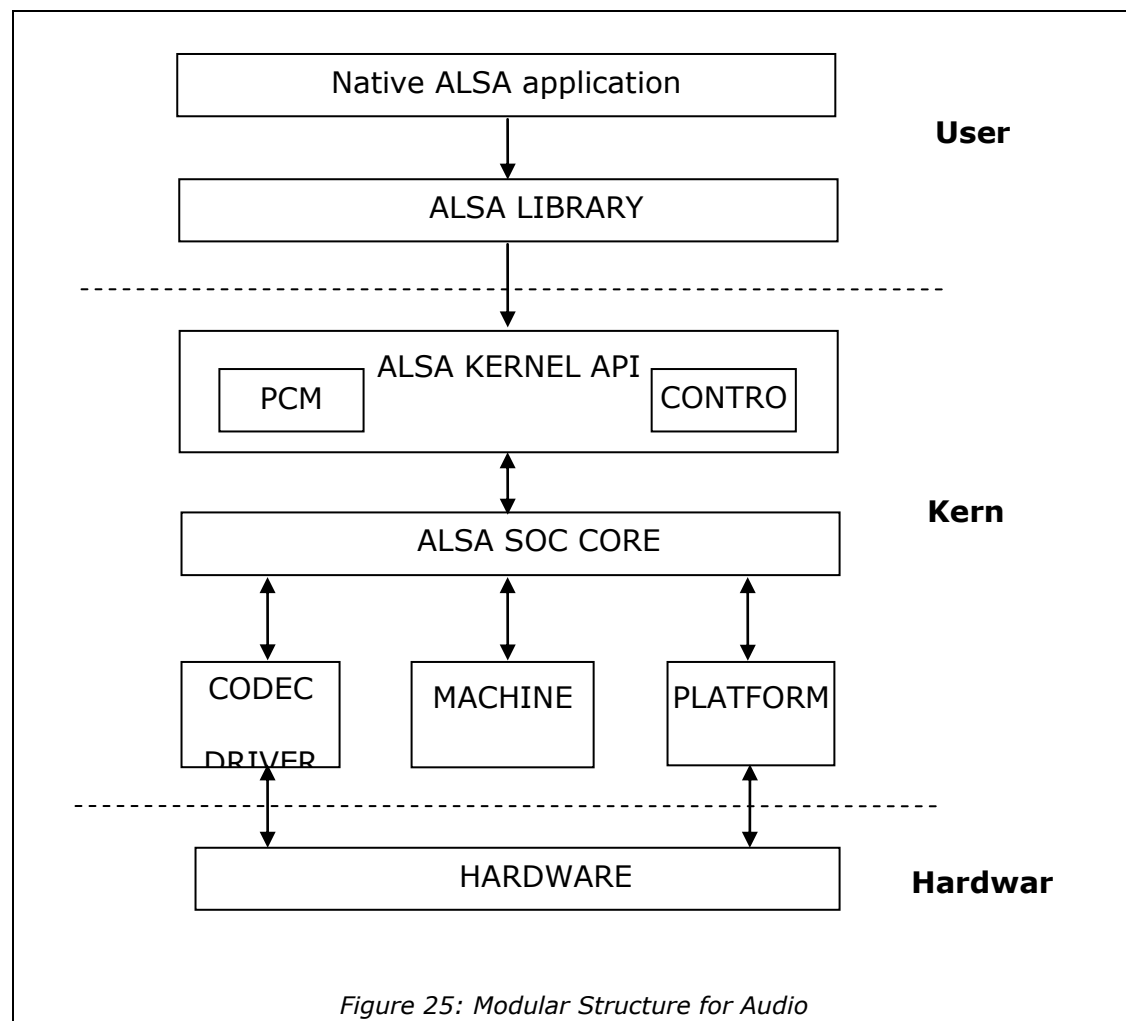
The da8xx-fb LCD driver can be used by making enhancements to the LCD_VERSION2 code. This update in LCDC version can be detected by reading the PID register.

Kernel Driver reference paths:

 linux-3.1.0-psp04.06.00.03.sdk/drivers/video/

 linux-3.1.0-psp04.06.00.03.sdk/drivers/video/da8xx-fb.c

5.5.4 Audio in/out



ASoC basically splits an embedded audio system into three components:

- **Codec driver:** The codec driver is platform independent and contains audio controls, audio interface capabilities, codec dapm definition and codec IO functions.
- **Platform driver:** The platform driver contains the audio dma engine and audio interface drivers (e.g. I2S, AC97, PCM) for that platform.
- **Machine driver:** The machine driver handles any machine specific controls and audio events i.e. turning on an amplifier at the start of playback.

Kernel Driver reference paths:

linux-3.1.0-psp04.06.00.03.sdk/sound/soc/

linux-3.1.0-psp04.06.00.03.sdk/sound/soc/davinci/davinci-evm.c

linux-3.1.0-psp04.06.00.03.sdk/sound/soc/codecs/sgtl5000.c

5.6 Driver Development

5.6.1 gpio_keys Driver

Device Definition

linux-3.1.0-psp04.06.00.03.sdk/arch/arm/mach-omap2/board-am335xevm.c

Setup **GPIO 1.30** as a “menu” key, with a return value of “**KEY_F1**”, triggered by a low level, and **GPIO 1.31** as a “back” key, with a return value of “**KEY_ESC**”, triggered by a low level.

The structure template is shown below. **GPIO 0.22** is used as a “home” key, with a return value of “**KEY_HOME**”, triggered by a low level.

```
static struct gpio_keys_button gpio_key_buttons[] = {
    {
        .code           = KEY_F1,
        .gpio           = GPIO_TO_PIN(1, 30),
        .active_low     = true,
        .desc           = "menu",
        .type           = EV_KEY,
        // .wakeup       = 1,
    },
    {
        .code           = KEY_ESC,
        .gpio           = GPIO_TO_PIN(1, 31),
        .active_low     = true,
        .desc           = "back",
        .type           = EV_KEY,
        // .wakeup       = 1,
    },
    {
        .code           = KEY_HOME,
        .gpio           = GPIO_TO_PIN(0, 22),
        .active_low     = true,
        .desc           = "home",
    }
}
```

```

        .type           = EV_KEY,
//        .wakeup         = 1,
    },

static struct gpio_keys_platform_data gpio_key_info = {
    .buttons           = gpio_key_buttons,
    .nbuttons          = ARRAY_SIZE(gpio_key_buttons),
};

static struct platform_device gpio_keys = {
    .name      = "gpio-keys",
    .id        = -1,
    .dev       = {
        .platform_data = &gpio_key_info,
    },
};

```

GPIO pinmux Configuration

Setup GPIO 1.30, GPIO1.31 and GPIO0.22 as M7(GPIO mode) and IEM (Input enable).

linux-3.1.0-psp04.06.00.03.sdk/arch/arm/mach-omap2/board-am335xev
m.c

```

static struct pinmux_config gpio_keys_pin_mux[] = {
    {"gpmc_csn1.gpio1_30", OMAP_MUX_MODE7 | AM33XX_PIN_INPUT},
    {"gpmc_csn2.gpio1_31", OMAP_MUX_MODE7 | AM33XX_PIN_INPUT},
    {"gpmc_ad8.gpio0_22",  OMAP_MUX_MODE7 | AM33XX_PIN_INPUT},
    {NULL, 0},
};

```

Driver Design

linux-3.1.0-psp04.06.00.03.sdk/drivers/input/keyboard/gpio_keys.c

Structure for platform_driver_register to register gpio_keys driver:

```

static struct platform_driver gpio_keys_device_driver = {
    .probe      = gpio_keys_probe,
    .remove     = __devexit_p(gpio_keys_remove),
    .driver     = {
        .name    = "gpio-keys",
    },
};

```

```
        .owner = THIS_MODULE,
        .pm     = &gpio_keys_pm_ops,
        .of_match_table = gpio_keys_of_match,
    }
};

static int __init gpio_keys_init(void)
{
    return platform_driver_register(&gpio_keys_device_driver);
}

static void __exit gpio_keys_exit(void)
{
    platform_driver_unregister(&gpio_keys_device_driver);
}

late_initcall(gpio_keys_init);
module_exit(gpio_keys_exit);

MODULE_LICENSE("GPL");
MODULE_AUTHOR("Phil Blundell <pb@handhelds.org>");
MODULE_DESCRIPTION("Keyboard driver for GPIOs");
MODULE_ALIAS("platform:gpio-keys");
```

Structure for `input_register_device` to register input driver:

```
static int __devinit gpio_keys_probe(struct platform_device *pdev)
{
    ...

    input = input_allocate_device();
    ...

    for (i = 0; i < pdata->nbuttons; i++) {
        struct gpio_keys_button *button = &pdata->buttons[i];
        struct gpio_button_data *bdata = &ddata->data[i];
        unsigned int type = button->type ?: EV_KEY;

        bdata->input = input;
        bdata->button = button;

        error = gpio_keys_setup_key(pdev, bdata, button);
        if (error)
            goto fail2;

        if (button->wakeup)
```

```
wakeup = 1;

    input_set_capability(input, type, button->code);
}
error = sysfs_create_group(&pdev->dev.kobj, &gpio_keys_attr_group);
if (error) {
    dev_err(dev, "Unable to export keys/switches, error: %d\n",
            error);
    goto fail2;
}

error = input_register_device(input);
if (error) {
    dev_err(dev, "Unable to register input device, error: %d\n",
            error);
    goto fail3;
}

...
```

Apply GPIO and setup the GPIO as the input, register GPIO interrupt.

```
static int __devinit gpio_keys_setup_key(struct platform_device *pdev,
                                         struct gpio_button_data *bdata,
                                         struct gpio_keys_button *button)
{
    const char *desc = button->desc ? button->desc : "gpio_keys";
    struct device *dev = &pdev->dev;
    unsigned long irqflags;
    int irq, error;

    setup_timer(&bdata->timer, gpio_keys_timer, (unsigned long)bdata);
    INIT_WORK(&bdata->work, gpio_keys_work_func);

    error = gpio_request(button->gpio, desc);
    if (error < 0) {
        dev_err(dev, "failed to request GPIO %d, error %d\n",
                button->gpio, error);
        goto fail2;
    }

    error = gpio_direction_input(button->gpio);
    if (error < 0) {
        dev_err(dev, "failed to configure"
                " direction for GPIO %d, error %d\n",
```

```
        button->gpio, error);
    goto fail3;
}

if (button->debounce_interval) {
    error = gpio_set_debounce(button->gpio,
                              button->debounce_interval * 1000);
    /* use timer if gpiolib doesn't provide debounce */
    if (error < 0)
        bdata->timer_debounce = button->debounce_interval;
}

irq = gpio_to_irq(button->gpio);
if (irq < 0) {
    error = irq;
    dev_err(dev, "Unable to get irq number for GPIO %d, error %d\n",
            button->gpio, error);
    goto fail3;
}

irqflags = IRQF_TRIGGER_RISING | IRQF_TRIGGER_FALLING;
/*
 * If platform has specified that the button can be disabled,
 * we don't want it to share the interrupt line.
 */
if (!button->can_disable)
    irqflags |= IRQF_SHARED;

error = request_threaded_irq(irq, NULL, gpio_keys_isr, irqflags, desc,
bdata);
if (error < 0) {
    dev_err(dev, "Unable to claim irq %d; error %d\n",
            irq, error);
    goto fail3;
}

return 0;

fail3:
    gpio_free(button->gpio);
fail2:
    return error;
}
```

Interrupt Handling,

Button is pressed and an interrupt is generated:

```
static irqreturn_t gpio_keys_isr(int irq, void *dev_id)
{
...
schedule_work(&bdata->work);
...
}

static void gpio_keys_work_func(struct work_struct *work)
{
...
gpio_keys_report_event(bdata);
...
}

static void gpio_keys_report_event(struct gpio_button_data *bdata)
{
    struct gpio_keys_button *button = bdata->button;
    struct input_dev *input = bdata->input;
    unsigned int type = button->type ?: EV_KEY;
    int state = (gpio_get_value(button->gpio) ? 1 : 0) ^ button->active_low;

    input_event(input, type, button->code, !!state);
    input_sync(input);
}
```

5.6.2 Driver for the GPIO_Leds

Device Definition

 linux-3.1.0-psp04.06.00.03.sdk/arch/arm/mach-omap2/board-am335xevm.c

The kernel configurations are: sys_led (GPIO1.26) and usr_led (GPIO1.27), where a low level enables:

```
static struct gpio_led gpio_leds[] = {
    {
        .name          = "sys_led",
        .default_trigger = "heartbeat",
        .gpio           = GPIO_TO_PIN(1, 26),
    },
    {
        .name          = "user_led",
        .gpio           = GPIO_TO_PIN(1, 27),
    },
};

static struct gpio_led_platform_data gpio_led_info = {
    .leds          = gpio_leds,
    .num_leds       = ARRAY_SIZE(gpio_leds),
};

static struct platform_device leds_gpio = {
    .name = "leds-gpio",
    .id   = -1,
    .dev  = {
        .platform_data = &gpio_led_info,
    },
};
```

GPIO pinmux Setup

linux-3.1.0-psp04.06.00.03.sdk/arch/arm/mach-omap2/board-am335xevm.c

Configure GPIO 1.26 and GPIO 1.27 as M7 (MODE 7 = GPIO) and IDIS (Input not allowed)

```
static struct pinmux_config gpio_led_pin_mux[] = {
    {"gpmc_a10.gpio1_26", OMAP_MUX_MODE7 | AM33XX_PIN_OUTPUT},
```

```

    {"gpmc_a11.gpio1_27",    OMAP_MUX_MODE7 | AM33XX_PIN_OUTPUT},
    {NULL, 0},
};

```

Driver design:

linux-3.1.0-psp04.06.00.03.sdk/drivers/leds/leds-gpio.c

Structure for platform_driver_register to register gpio_leds.

```

static struct platform_driver gpio_led_driver = {
    .probe      = gpio_led_probe,
    .remove     = __devexit_p(gpio_led_remove),
    .driver     = {
        .name    = "leds-gpio",
        .owner   = THIS_MODULE,
        .of_match_table = of_gpio_leds_match,
    },
};

MODULE_ALIAS("platform:leds-gpio");

static int __init gpio_led_init(void)
{
    return platform_driver_register(&gpio_led_driver);
}

static void __exit gpio_led_exit(void)
{
    platform_driver_unregister(&gpio_led_driver);
}

module_init(gpio_led_init);
module_exit(gpio_led_exit);

MODULE_AUTHOR("Raphael Assenat <raph@8d.com>, Trent Piepho
<tpiepho@freescale.com>");
MODULE_DESCRIPTION("GPIO LED driver");
MODULE_LICENSE("GPL");

```

Apply GPIO and call led_classdev_register to register led_classdev.

```

static int __devinit gpio_led_probe(struct platform_device *pdev)

```

```

{
...
    if (pdata && pdata->num_leds) {
        priv = kzalloc(sizeof_gpio_leds_priv(pdata->num_leds),
                        GFP_KERNEL);

        if (!priv)
            return -ENOMEM;

        priv->num_leds = pdata->num_leds;
        for (i = 0; i < priv->num_leds; i++) {
            ret = create_gpio_led(&pdata->leds[i],
                                &priv->leds[i],
                                &pdev->dev, pdata->gpio_blink_set);

            if (ret < 0) {
                /* On failure: unwind the led creations */
                for (i = i - 1; i >= 0; i--)
                    delete_gpio_led(&priv->leds[i]);
                kfree(priv);
                return ret;
            }
        }
    }
...
}

static int __devinit create_gpio_led(const struct gpio_led *template,
                                     struct gpio_led_data *led_dat, struct device *parent,
                                     int (*blink_set)(unsigned, unsigned long *, unsigned long *))
{
...

    ret = gpio_request(template->gpio, template->name);

...

    ret = gpio_direction_output(led_dat->gpio, led_dat->active_low ^ state);

...

    ret = led_classdev_register(parent, &led_dat->cdev);

...
}

```

User can access the brightness file in the directory `/sys/class/leds/xxx/brightness`, and call the function `gpio_led_set` to configure LED states.

```

static void gpio_led_set(struct led_classdev *led_cdev,
                        enum led_brightness value)
{

```

```
...  
    gpio_set_value(led_dat->gpio, level);  
}
```

5.7 Updating the System

5.7.1 Update of the TF Card System Image

When formatting the MMC/SD card it is highly recommended that the HP USB Disk Storage Format Tool is used. The software is available from:



<http://www.embest-tech.com/resource/download/HP-USB-Disk-Storage-Format-Tool.rar>.

1. Insert the TF card into the card reader of a PC.
2. Open the HP USB Disk Storage Format Tool, the following steps will be shown in detail:

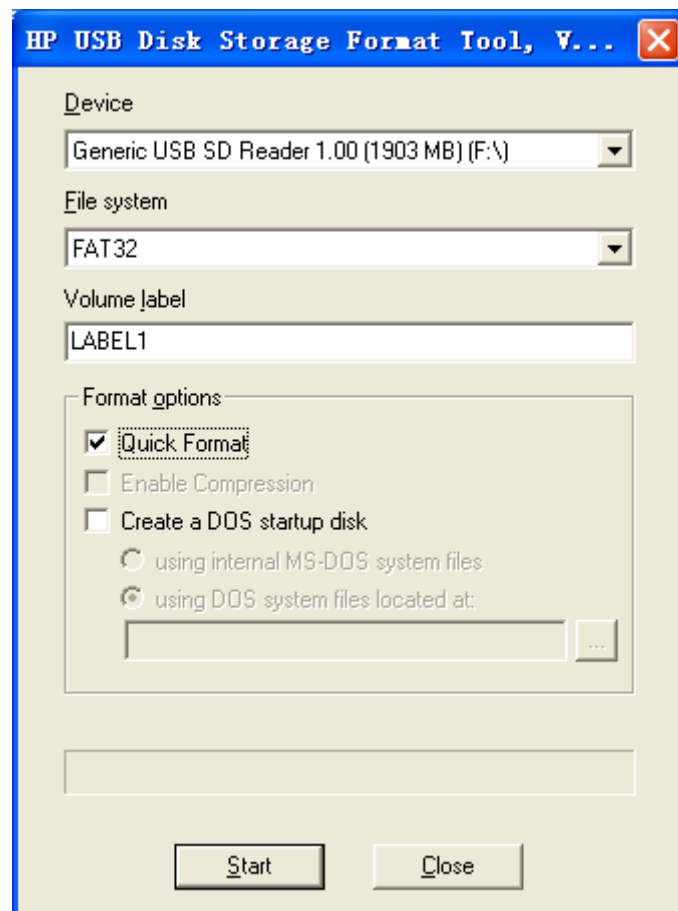


Figure 26: USB Disk Storage Format Tool

3. Select the file system "FAT32".

4. Click **Start**.

5. When formatting is completed, click **OK**.

Note:

 The HP USB Disk Storage Format Tool will clear partitions of the TF card.

 Use other format tool may cause the failure of the TF card booting

Updating Images

Copy all files from the directory linux/image to the TF card, and **rename uImage_xx** substituting the xx according to the display device to be used e.g. 4.3, 7, LVDS or VGA. Connect the TF card then power on and boot the system, the serial port information will be displayed as follows:

```
U-Boot SPL 2011.09-svn (Mar 02 2012 - 17:15:32)
Texas Instruments Revision detection unimplemented
Booting from MMC...
OMAP SD/MMC: 0
reading u-boot.img
reading u-boot.img

U-Boot 2011.09-svn (Mar 02 2012 - 17:15:32)

I2C:  ready
DRAM:  512 MiB
WARNING: Caches not enabled
Did not find a recognized configuration, assuming General purpose EVM in Profile
0 with Daughter board
NAND:  HW ECC Hamming Code selected
nand_get_flash_type: unknown NAND device: Manufacturer ID: 0xad, Chip ID: 0xd7
No NAND device found!!!
0 MiB
MMC:  OMAP SD/MMC: 0
*** Warning - readenv() failed, using default environment

Net:  cpsw
Hit key to stop autoboot:  0
```

```
SD/MMC found on device 0
reading uEnv.txt

** Unable to read "uEnv.txt" from mmc 0:1 **
reading uImage

2993120 bytes read
reading ramdisk.gz

12132646 bytes read
## Booting kernel from Legacy Image at 80007fc0 ...
   Image Name:   Linux-3.1.0
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    2993056 Bytes = 2.9 MiB
   Load Address: 80008000
   Entry Point:  80008000
   Verifying Checksum ... OK
   XIP Kernel Image ... OK
OK

Starting kernel ...

Uncompressing Linux... done, booting the kernel.
Linux version 3.1.0 (luofc@TIOP) (gcc version 4.3.3 (Sourcery G++ Lite 2009q1-203)
) #28 Mon Mar 5 11:04:25 CST 2012
.....
.....
RAMDISK: gzip image found at block 0
VFS: Mounted root (ext2 filesystem) on device 1:0.
Freeing init memory: 240K
INIT: version 2.86 booting
Starting udevdevd (623): /proc/623/oom_adj is deprecated, please use
/proc/623/oom_score_adj instead.
tar: removing leading '/' from member names

Remounting root file system...
mount: mounting /dev/root on / failed: Invalid argument
mount: mounting /dev/root on / failed: Invalid argument
root: mount: mounting rootfs on / failed: No such file or directory
Setting up IP spoofing protection: rp_filter.
Configuring network interfaces... udhcpc (v1.11.3) started
Sending discover...
udhcpc: sendto: Network is down
Sending discover...
```

```

udhcpc: sendto: Network is down
Sending discover...
udhcpc: sendto: Network is down
No lease, failing
done.
Tue Jan 27 08:47:00 UTC 2009
INIT: Entering runlevel: 5
Starting syslogd/klogd: done

.-----.
|       |               .-.
| | | |-----.-----.| | | .-----.-----
| | | | | _ | ---' | '---. | .-' | | | |
| | | | | | | --- || --' | | | ' | | | |
'-----'-----'-----'-----'-----'-----'
               -' |
               '----'

The Angstrom Distribution DevKit8600 tty00

Angstrom 2008.1-test-20090127 DevKit8600 tty00

DevKit8600 login: (Just input "root")

```

HyperTerminal displays above information to indicate that it has successfully booted the Linux system from the TF card.

5.7.2 Updating NAND Flash

Updating of the NAND boot image is completed with the aid of u-boot. Regardless of whether NAND Flash currently contains data, the u-boot of the TF card can be used to update the NAND Flash image.

1. Format the TF card to the FAT or FAT32 file system using the HP USB Disk Storage Format Tool
2. Copy the **MLO, u-boot.img, uImage_xx and ubi.img** substituting the xx according to the display device to be used e.g. 4.3, 7, LVDS or VGA.
3. Insert the TF card with the system images into the development board, power on and boot it, and press any key on the PC keyboard to enter u-boot according to the following prompts:

```
U-Boot SPL 2011.09-svn (Mar 02 2012 - 17:15:32)
Texas Instruments Revision detection unimplemented
Booting from MMC...
OMAP SD/MMC: 0
reading u-boot.img
reading u-boot.img

U-Boot 2011.09-svn (Mar 02 2012 - 17:15:32)

I2C:  ready
DRAM:  512 MiB
WARNING: Caches not enabled
Did not find a recognized configuration, assuming General purpose EVM in Profile
0 with Daughter board
NAND:  HW ECC Hamming Code selected
nand_get_flash_type: unknown NAND device: Manufacturer ID: 0xad, Chip ID: 0xd7
No NAND device found!!!
0 MiB
MMC:  OMAP SD/MMC: 0
*** Warning - readenv() failed, using default environment

Net:  cpsw

Hit any key to stop autoboot:  0 (Here press any key to enter u-boot)
After entering the u-boot command line, input "run updatesys" from the PC keyboard,
to start to update the system automatically:
Devkit8600# run updatesys

NAND erase.chip: device 0 whole chip
Erasing at 0x7fe0000 -- 100% complete.
OK
reading MLO

38151 bytes read
HW ECC BCH8 Selected

NAND write: device 0 offset 0x0, size 0x9507
 38151 bytes written: OK
reading u-boot.img

232456 bytes read
HW ECC BCH8 Selected

NAND write: device 0 offset 0x80000, size 0x38c08
```

```
232456 bytes written: OK
reading uImage

2984304 bytes read
HW ECC BCH8 Selected

NAND write: device 0 offset 0x280000, size 0x2d8970
2984304 bytes written: OK
reading ubi.img

20447232 bytes read
HW ECC BCH8 Selected

NAND write: device 0 offset 0x780000, size 0x1380000
20447232 bytes written: OK
```

4. At this time, a flickering of the LED on the board indicates that the update has been finished; you just need to remove the TF card and reboot the board.

5.8 Various Test Scenarios

5.8.1 LED Testing

On the board, D48 is the System heartbeat lamp and D49 is the user's led lamp.

The following operation can be carried out in HyperTerminal to control both LEDs:

1. System heartbeat lamp:

```
root@DevKit8600:~# echo 1 > /sys/class/leds/sys_led/brightness
root@DevKit8600:~# echo 0 > /sys/class/leds/sys_led/brightness
```

2. User's led lamp:

```
root@DevKit8600:~# echo 1 > /sys/class/leds/user_led/brightness
root@DevKit8600:~# echo 0 > /sys/class/leds/user_led/brightness
```

5.8.2 KEYPAD Testing

The board has three user configurable keys: BACK, MENU and HOME; users can perform the following testing:

```
root@DevKit8600:~# evtest /dev/input/event1
Input driver version is 1.0.1
```

```
Input device ID: bus 0x19 vendor 0x1 product 0x1 version 0x100
Input device name: "gpio-keys"
Supported events:
  Event type 0 (Sync)
  Event type 1 (Key)
    Event code 1 (Esc)
    Event code 59 (F1)
    Event code 102 (Home)
Testing ... (interrupt to exit)
Event: time 1233046035.953970, type 1 (Key), code 102 (Home), value 1
Event: time 1233046035.953975, ----- Report Sync -----
Event: time 1233046036.095752, type 1 (Key), code 102 (Home), value 0
Event: time 1233046036.095753, ----- Report Sync -----
Event: time 1233046037.867785, type 1 (Key), code 59 (F1), value 1
Event: time 1233046037.867788, ----- Report Sync -----
Event: time 1233046038.000793, type 1 (Key), code 59 (F1), value 0
Event: time 1233046038.000795, ----- Report Sync -----
Event: time 1233046038.854748, type 1 (Key), code 1 (Esc), value 1
Event: time 1233046038.854751, ----- Report Sync -----
Event: time 1233046039.022872, type 1 (Key), code 1 (Esc), value 0
```

Note:

 Press CONTROL+C to quit the test.

5.8.3 Touch Screen Testing

This testing requires Linux to boot from NAND Flash

1. Run the following command to test the touch screen:

```
root@DevKit8600: # ts_calibrate
```

Then follow the LCD prompts, click the "+" icon 5 times to complete the calibration

2. Once calibration is complete, enter the following commands to begin the Touch Panel Test:

```
root@DevKit8600: # ts_test
```

Follow the LCD prompts to choose the draw point and draw line tests.

5.8.4 Backlight Testing

After entering the system, run the following command to test the backlight.

Backlight brightness setting range from 0~100, where a value of 100 is the brightest and a value of 0 turns off the backlight.

1. View the backlight brightness at the default level.

```
root@DevKit8600:~# cat /sys/class/backlight/pwm-backlight/brightness
50
```

2. Set the backlight brightness to 0

```
root@DevKit8600:~# echo 0 > /sys/class/backlight/pwm-backlight/brightness
root@DevKit8600:~# cat /sys/class/backlight/pwm-backlight/brightness
0
```

At this time the backlight is turned off, the screen goes black.

3. Set the backlight brightness to 100

```
root@DevKit8600:~# echo 100 > /sys/class/backlight/pwm-backlight/brightness
root@DevKit8600:~# cat /sys/class/backlight/pwm-backlight/brightness
100
```

At this time the backlight is set to the maximum

5.8.5 RTC Testing

The development board contains a hardware clock for the saving and synchronisation of the system time. Test can be made with the following steps:

1. Set the system time to March 22, 2012, 8:00 pm

```
root@DevKit8600: # date 032220002012
Thu Mar 22 20:00:00 UTC 2012
```

2. Write the system clock into the RTC

```
root@ DevKit8600: # hwclock -w
```

3. Read the RTC

```
root@ DevKit8600: # hwclock
Thu Mar 22 20:00:10 2012 0.000000 seconds
```

We can see that the RTC clock has been set as March 22, 2012; the system clock will be saved in the hardware clock.

4. Restart the system and enter the following commands to renew the system clock

```
root@DevKit8600: # hwclock -s
```

```
root@DevKit8600: # date
Thu Mar 22 20:01:30 2012 0.000000 seconds
```

We can see the system time is now set as hardware time.

Note:

-  RTC will halt up after turn off, this is a bug for the CPU, and TI had release the corrigendum, please refer to the <http://www.ti.com/lit/er/sprz360b/sprz360b.pdf>
-  The DevKit8600 Development board RTC requires a CR1220 battery. The user must provide this themselves.

5.8.6 TF Card Testing

1. After connecting the TF card, the system will mount the file system of the TF card under the directory /media automatically:

```
root@DevKit8600:~# cd /media/
root@DevKit8600:/media# ls
card      hdd      mmcblk0p1  ram      union
cf        mmc1     net        realroot
```

2. Enter the following command , this will allow you to see the contents of the TF card:

```
root@DevKit8600:/media# ls mmcblk0p1/
flash-uboot.bin      u-boot.bin          x-load.bin.ift_for_NAND
mlo                  uImage
ramdisk.gz           ubi.img
```

5.8.7 USB Devices Testing

During USB DEVICE testing, a USB mini B to USB A cable is used to connect the Mini-USB interface of the development board and the USB interface of the computer. The computer recognises the development board as a network device to facilitate ping communication between the two systems.

1. After connecting the system to the PC, the computer needs to be installed with Linux USB Ethernet driver. For instructions please refer to Driver Installation Of Linux USB Ethernet/RNDIS Gadget
2. The following commands are input into HyperTerminal:

```
root@DevKit8600:~# ifconfig usb0 192.168.1.115
root@DevKit8600:~# ifconfig
```

```
lo      Link encap:Local Loopback
        inet addr:127.0.0.1  Mask:255.0.0.0
        UP LOOPBACK RUNNING  MTU:16436  Metric:1
        RX packets:26 errors:0 dropped:0 overruns:0 frame:0
        TX packets:26 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:0
        RX bytes:2316 (2.2 KiB)  TX bytes:2316 (2.2 KiB)

usb0    Link encap:Ethernet  HWaddr 5E:C5:F6:D4:2B:91
        inet addr:192.168.1.115  Bcast:192.168.1.255  Mask:255.255.255.0
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        RX packets:253 errors:0 dropped:0 overruns:0 frame:0
        TX packets:43 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:35277 (34.4 KiB)  TX bytes:10152 (9.9 KiB)
```

3. After the development board has been configured, please click

- 🖱️ My Computer
- 🖱️ Network Neighbourhood
- 🖱️ Check Network Connection

A virtual network adapter will be added to the PC.

4. Right-click the virtual network adapter on the computer, left-click **Attributes**, then double-click to enter the “Internet Protocol (TCP/IP)” window. Here you can configure the IP address of the virtual network adapter:

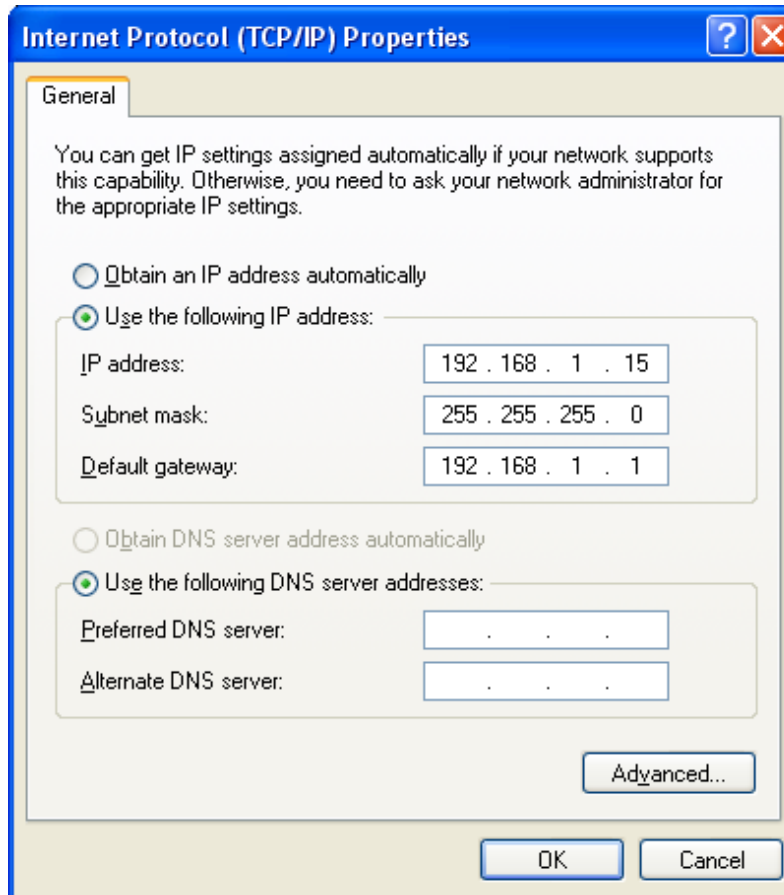


Figure 27: IP Settings

5. Use the ping command in the HyperTerminal window to test whether the settings of the development board are correct:

```
root@DevKit8600:~# ping 192.168.1.15
PING 192.168.1.15 (192.168.1.15): 56 data bytes
64 bytes from 192.168.1.15: seq=0 ttl=128 time=0.885 ms
64 bytes from 192.168.1.15: seq=1 ttl=128 time=0.550 ms
```

6. The above information indicates that the testing was successful.

Note:

 The IP address of the network adapter configured in OTG cannot be the same as that of Ethernet interface.

5.8.8 USB HOST Testing

1. After connecting a USB flash drive, the system will mount the file system of the USB flash disk under the directory /media automatically:

```

root@DevKit8600:~# cd /media/
root@ DevKit8600:/media# ls
card      hdd      mmcblk0p1  ram      sdal
cf        mmc1     net        realroot union

```

2. The contents of the USB flash disk will be seen after the following instruction is input:

```

root@DevKit8600:/media# ls sdal/
flash-uboot.bin      u-boot.bin          x-load.bin.ift_for_NAND
mlo                  uImage
ramdisk.gz           ubi.img

```

5.8.9 Audio Testing

The board has an audio input and output interface, and there are alsa-utils audio test tools in the file system, users can enter the following commands for a test:

1. Recording Test :

Plug in a microphone to test the recording functionality.

```

root@DevKit8600:~# arecord -t wav -c 1 -r 44100 -f S16_LE -v k
Recording WAVE 'k' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
Plug PCM: Hardware PCM card 0 'omap3evm' device 0 subdevice 0
Its setup is:
  stream      : CAPTURE
  access      : RW_INTERLEAVED
  format      : S16_LE
  subformat   : STD
  channels     : 2
  rate        : 44100
  exact rate   : 44100 (44100/1)
  msbits      : 16
  buffer_size  : 22052
  period_size  : 5513
  period_time  : 125011
  tstamp_mode  : NONE
  period_step  : 1
  avail_min    : 5513
  period_event : 0
  start_threshold : 1
  stop_threshold : 22052
  silence_threshold: 0
  silence_size : 0

```

```
boundary      : 1445199872
appl_ptr      : 0
hw_ptr        : 0
```

2. Playback Testing :

Plug in headphones to hear what you have just recorded.

```
root@DevKit8600:~# aplay -t wav -c 2 -r 44100 -f S16_LE -v k
Playing WAVE 'k' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
Plug PCM: Hardware PCM card 0 'omap3evm' device 0 subdevice 0
Its setup is:
  stream       : PLAYBACK
  access       : RW_INTERLEAVED
  format       : S16_LE
  subformat    : STD
  channels     : 2
  rate         : 44100
  exact rate   : 44100 (44100/1)
  msbits       : 16
  buffer_size  : 22052
  period_size  : 5513
  period_time  : 125011
  timestamp_mode : NONE
  period_step  : 1
  avail_min    : 5513
  period_event : 0
  start_threshold : 22052
  stop_threshold  : 22052
  silence_threshold: 0
  silence_size   : 0
  boundary      : 1445199872
  appl_ptr      : 0
  hw_ptr        : 0
```

5.8.10 Network Testing

1. Users can connect the board to the router or switch and enter the following commands for a test:

```
root@DevKit8600:~# ifconfig eth0 192.192.192.200
[root@DevKit8600 /]# ifconfig
eth0      Link encap:Ethernet  HWaddr D4:94:A1:8D:EB:25
          inet addr:192.192.192.200  Bcast:192.192.192.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:137 errors:0 dropped:4 overruns:0 frame:0
```

```

TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:13792 (13.4 KiB) TX bytes:0 (0.0 B)
Interrupt:40

lo      Link encap:Local Loopback
        inet addr:127.0.0.1  Mask:255.0.0.0
        UP LOOPBACK RUNNING  MTU:16436  Metric:1
        RX packets:0 errors:0 dropped:0 overruns:0 frame:0
        TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:0
        RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

[root@DevKit8600 /]# ping 192.192.192.170
PING 192.192.192.170 (192.192.192.170): 56 data bytes
64 bytes from 192.192.192.170: seq=0 ttl=128 time=4.486 ms
64 bytes from 192.192.192.170: seq=1 ttl=128 time=0.336 ms

```

2. The above information indicates that the testing is successful.

5.8.11 CAN Testing

The DevKit8600 can be used as a CAN device. The following figure shows the connection principle and can be used to find the corresponding pins to connect the DevKit8600 CAN interface to another CAN device.

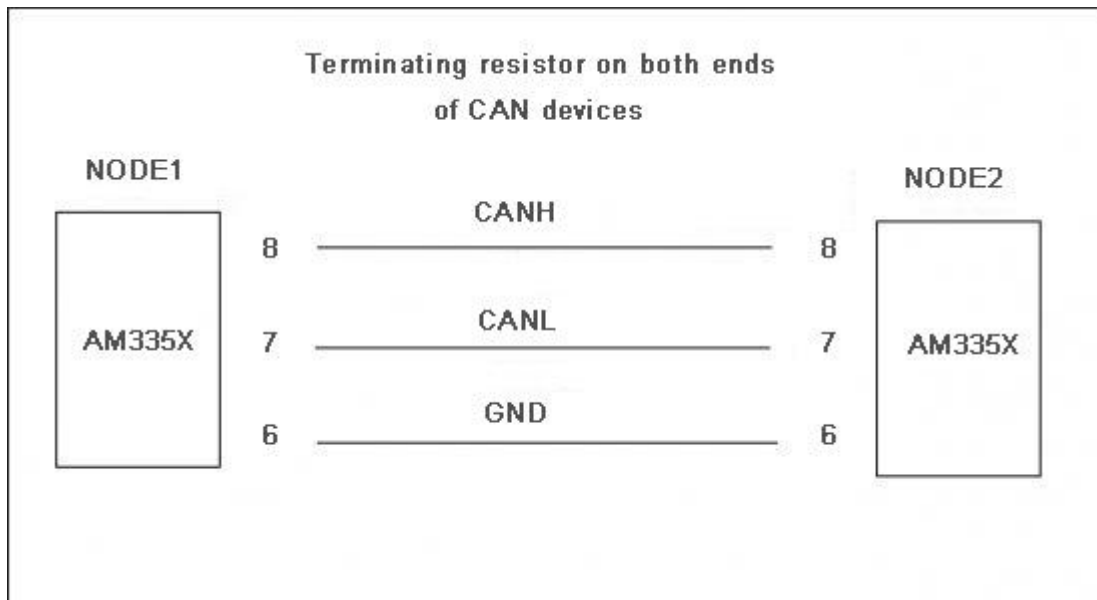


Figure 28: CAN Connections

The test method is shown below:

1. Both DevKit8600 and another CAN device have their baud rate set to 125KBPS, and their CAN devices enabled.

```
root@DevKit8600:~# canconfig can0 bitrate 125000 ctrlmode triple-sampling on
root@DevKit8600:~# canconfig can0 start
```

2. Once the DevKit8600 and another CAN device are connected, enter the following command to send data packets

```
root@DevKit8600:~# cansend can0 -i 0x10 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88
```

Note:

-  This command will only send data once, to resend data you need to re-enter the command.
-  Ensure that the other device is in a receiving state and it will then print the information sent

3. Receive data packets

```
root@DevKit8600:~# candump can0
```

Execute the command; the terminal will print the received data.

4. Close the device.

```
root@DevKit8600:~# canconfig can0 stop
```

According to the above command, the devices can send and receive data to each other; what's more, it is possible to set a different baud rate for communication. You must close the CAN device before modification of the baud rate.

The baud rate can be set to any of the values below:

- 25KBPS (250000)
- 50KBPS (50000)
- 125KBPS (125000)
- 500KBPS (500000)
- 650KBPS (650000)
- 1MKBPS (1000000)

Using the above baud rates, the CAN device can communicate normally. Other baud rates may be chosen by the user if desired.

Note:

 The baud rates of the two CAN devices need to be the same.

5.8.12 RS485 Testing

The DevKit8600 can be used as an RS485 device. The following figure shows the connection principle and can be used to find the corresponding pins to connect the DevKit8600 CAN interface to another CAN device.

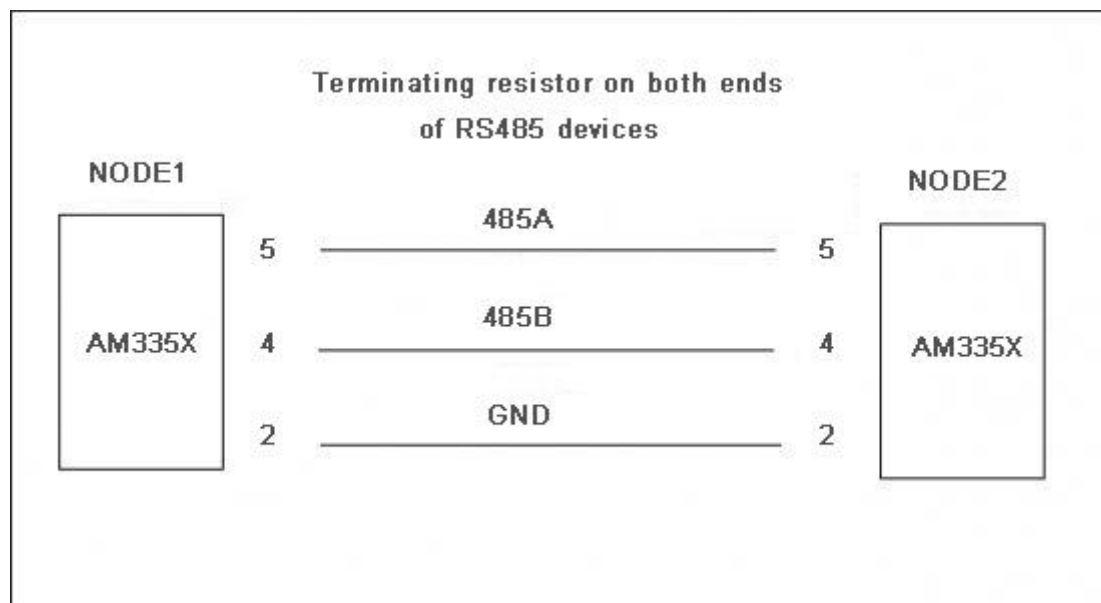


Figure 29: RS485 Connections

Note:

 RS485 only supports half-duplex communication, i.e. communication works in both directions, but only in one direction at a time (not simultaneously).

Copy 485_test(linux\example\rs485_test) to the TF card, Insert the TF card into the Devkit8600 TF slot and run the following command:

```
root@DevKit8600:~# cd /media/mmcblk0p1/
root@DevKit8600:/media/mmcblk0p1# ./485_test -d /dev/ttyO1 -b 115200
*****
485 TEST
*****
```

```
Select 1 : Send a message
Select 2 : Receive messages
>
```

On one end to send information:

```
Select 1 : Send a message
Select 2 : Receive messages
>1

    Please enter the information to be sent off!
-----gpio0_13 set to 0
115200
message = 115200
len = 6
Information is sent.....
Select 3 : Stop Send
>
```

And on another end to receive:

```
Select 1 : Send a message
Select 2 : Receive messages
>2

-----gpio0_13 set to 1
Select 3 : Stop Receive
>

RECV: 6 : 115200
RECV: 6 : 115200
RECV: 6 : 115200
RECV: 6 : 115200
```

To stop receiving information:

```
Select 3 : Stop Receive
>3
```

5.8.13 Serial port testing

Because UART1 (Compatible with RS485) and UART3 (Compatible with WiFi/Bluetooth) are occupied, this chapter will show you how to test UART2.

After short circuiting the UART2_RX_3V3 pin and the UART2_TX_3V3 pin, copy the file `uart_test` (linux\example\uart_test) to the TF card, insert the TF card to the Devkit8600 TF slot, and input the commands as below:

```
root@DevKit8600:~# cd /media/mmcblk0p1/
root@DevKit8600:/media/mmcblk0p1# ./uart_test -d /dev/ttyO2 -b 115200
```

Receiving the following information indicates the test was successful.

```
/dev/ttyO2 SEND: 1234567890
/dev/ttyO2 RECV 10 total
/dev/ttyO2 RECV: 1234567890
/dev/ttyO2 SEND: 1234567890
/dev/ttyO2 RECV 10 total
/dev/ttyO2 RECV: 1234567890
/dev/ttyO2 SEND: 1234567890
/dev/ttyO2 RECV 10 total
/dev/ttyO2 RECV: 1234567890
/dev/ttyO2 SEND: 1234567890
/dev/ttyO2 RECV 10 total
/dev/ttyO2 RECV: 1234567890
/dev/ttyO2 SEND: 1234567890
/dev/ttyO2 RECV 10 total
/dev/ttyO2 RECV: 1234567890
```

5.8.14 WiFi Testing

To test the connection to a non-encrypted wireless router:

1. Enable WiFi

```
root@DevKit8600:~# ifconfig wlan0 up
wl1271: firmware booted (Rev 6.1.5.50.74)
```

2. Scan for a WiFi router

```
root@DevKit8600:~# iwlist wlan0 scan
wlan0    Scan completed :
    Cell 01 - Address: 94:0C:6D:17:0A:BC
            Channel:1
            Frequency:2.412 GHz (Channel 1)
            Quality=44/70  Signal level=-66 dBm
            Encryption key:off
            ESSID:"TIOP"
            Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s
            Mode:Master
            Extra:tsf=0000000023f51ee7
            Extra: Last beacon: 560ms ago
            IE: Unknown: 000454494F50
            IE: Unknown: 010482848B96
            IE: Unknown: 030101
.....
```

3. Connect to a WiFi router

```
root@DevKit8600:~# iwconfig wlan0 essid TIOP
```

Association completed.

4. Testing:

```
root@DevKit8600:~# ifconfig wlan0 192.192.192.215
root@DevKit8600:~# ping 192.192.192.90
PING 192.192.192.90 (192.192.192.90): 56 data bytes
64 bytes from 192.192.192.90: seq=0 ttl=64 time=32.260 ms
64 bytes from 192.192.192.90: seq=1 ttl=64 time=20.662 ms
64 bytes from 192.192.192.90: seq=2 ttl=64 time=20.419 ms
```

To test the connection to a WEP encrypted wireless router:

1. Enable WiFi:

```
root@DevKit8600:~# ifconfig wlan0 up
wl1271: firmware booted (Rev 6.1.5.50.74)
```

2. Scan WiFi router

```
root@DevKit8600:~# iwlist wlan0 scan
wlan0    Scan completed :
    Cell 01 - Address: 94:0C:6D:17:0A:BC
            Channel:1
            Frequency:2.412 GHz (Channel 1)
            Quality=19/70  Signal level=-91 dBm
            Encryption key:on // encryption enabled
            ESSID:"TIOP"
            Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s
            Mode:Master
            Extra:tsf=000000141698b596
            Extra: Last beacon: 33540ms ago
            IE: Unknown: 000454494F50
            IE: Unknown: 010482848B96
            IE: Unknown: 030101
```

3. Connect to WiFi router

```
root@DevKit8600:~# iwconfig wlan0 essid TIOP key s:abcde
root@DevKit8600:~# ifconfig wlan0 down
wl1271: down
root@DevKit8600:~# ifconfig wlan0 up
wl1271: firmware booted (Rev 6.1.5.50.74)
Association completed.
```

Note:

-  The **iwconfig wlan0 essid TIOP key s:abcde** command indicates that the connected wireless router name is TIOP, the KEY format is ASCII characters and the KEY is "abcde". Connection to the wireless router is "TIOP", if the key format of WEP router used is hexadecimal characters and the key is: 0123456789, the WEP encrypted wireless router connect command is:
iwconfig wlan0 ESSID TIOP key 0123-4567-89
-  Before the test, you must use the **ifconfig wlan0 down** to turn off the WiFi device, use the **ifconfig wlan0 up** command to turn it on
-  The DevKit8600 is unable to connect to a WiFi device that uses encryption more complex than WEP

4. Testing

```
root@DevKit8600:~# ifconfig wlan0 192.192.192.216
root@DevKit8600:~# ping 192.192.192.90
PING 192.192.192.90 (192.192.192.90): 56 data bytes
64 bytes from 192.192.192.90: seq=0 ttl=64 time=32.260 ms
64 bytes from 192.192.192.90: seq=1 ttl=64 time=20.662 ms
```

5.8.15 BT Testing

1. Enable BT:

```
root@DevKit8600:~# insmod
/lib/modules/3.1.0/kernel/drivers/bt_enable/gpio_en.ko

Gpio value is :23
WL1271: BT Enable
```

2. Configure BT:

```
root@DevKit8600:~# hciattach /dev/ttyO3 texas 115200
Found a Texas Instruments' chip!
Firmware file : /lib/firmware/TIInit_7.2.31.bts
Loaded BTS script version 1
texas: changing baud rate to 921600, flow control to 1
Device setup complete
```

3. Testing :

```
root@DevKit8600:~# hciconfig hci0 up
root@DevKit8600:~# hciotool scan
```

Scanning ...

00:12:FE:B7:75:A0

Lenovo-TD80t

5.8.16 CDMA8000-U module

If you have a CDMA8000-U camera module from Embest then you can download the module material from the below link:

<http://www.timll.com/chinese/uploadFile/cdma8000.rar>

5.8.17 WCDMA8000-U module

If you have a WCDMA8000-U camera module from Embest then you can download the module material from the below link:

<http://www.timll.com/chinese/uploadFile/WCDMA8000-110113.zip>

5.9 Demo

5.9.1 Demonstration of the Android System

The DevKit8600 provides an Android system demonstration, please follow the steps below:

1. Copy all files from the directory CD\linux\demo\android\image to the TF card, rename the corresponding file uImage_xx substituting the xx according to the display device to be used e.g. 4.3, 7, LVDS or VGA.
2. Insert the TF card in the development board and power it on; the HyperTerminal window will display the following information:

```
CCCCCCCC
U-Boot SPL 2011.09-svn (May 22 2012 - 11:19:00)
Texas Instruments Revision detection unimplemented
Booting from MMC...
OMAP SD/MMC: 0
reading u-boot.img
reading u-boot.img

U-Boot 2011.09-svn (May 24 2012 - 11:17:39)

I2C:  ready
DRAM:  512 MiB
WARNING: Caches not enabled
```

```
Did not find a recognized configuration, assuming General purpose EVM in Profile
0 with Daughter board
NAND: HW ECC Hamming Code selected
512 MiB
MMC: OMAP SD/MMC: 0
*** Warning - bad CRC, using default environment

NAND erase.chip: device 0 whole chip
Skipping bad block at 0x03620000
Erasing at 0x1ffe0000 -- 100% complete.
OK
reading MLO

38167 bytes read
HW ECC BCH8 Selected

NAND write: device 0 offset 0x0, size 0x9517
 38167 bytes written: OK
reading flash-uboot.img

230148 bytes read
HW ECC BCH8 Selected

NAND write: device 0 offset 0x80000, size 0x38304
 230148 bytes written: OK
reading uImage

2709040 bytes read
HW ECC BCH8 Selected

NAND write: device 0 offset 0x280000, size 0x295630
 2709040 bytes written: OK
reading ubi.img

72744960 bytes read
SW ECC selected

NAND write: device 0 offset 0x780000, size 0x4560000
Skip bad block 0x03620000
```

3. The LED lamp on the board will flicker after programming is finished, at this time, please remove the TF card.

4. Power the system on again and boot the Android operating system.

5.9.2 Demonstration of the TISDK System

1. Format a TF card into two partitions (please refer to Appendix 3: Linux Boot Disk Format for detailed instructions)
2. Insert the CD and the TF card into the PC, then enter the following commands in the terminal window of Ubuntu (use the commands appropriate for the LCD screen size).

For 4.3" LCD

```
cp /media/cdrom/linux/demo/tisdk/image/MLO /media/LABEL1
cp /media/cdrom/linux/demo/tisdk/image/u-boot.img /media/LABEL1
cp /media/cdrom/linux/demo/tisdk/image/uImage_4.3 /media/LABEL1/uImage
rm -rf /media/LABEL2/*
sudo tar jxvf linux/demo/dvSDK/image/tisdk-rootfs-am335x-evm.tar.gz -C
/media/LABEL2
sync
umount /media/LABEL1
umount /media/LABEL2
```

For 7" LCD

```
cp /media/cdrom/linux/demo/tisdk/image/MLO /media/LABEL1
cp /media/cdrom/linux/demo/tisdk/image/u-boot.img /media/LABEL1
cp /media/cdrom/linux/demo/tisdk/image/uImage_7 /media/LABEL1/uImage
rm -rf /media/LABEL2/*
sudo tar xvf
/media/cdrom/linux/demo/tisdk/image/tisdk-rootfs-am335x-evm.tar.gz -C
/media/LABEL2
sync
umount /media/LABEL1
umount /media/LABEL2
```

For LVDS (10.4")

```
cp /media/cdrom/linux/demo/tisdk/image/MLO /media/LABEL1
cp /media/cdrom/linux/demo/tisdk/image/u-boot.img /media/LABEL1
cp /media/cdrom/linux/demo/tisdk/image/uImage_LVDS /media/LABEL1/uImage
rm -rf /media/LABEL2/*
sudo tar xvf
/media/cdrom/linux/demo/tisdk/image/tisdk-rootfs-am335x-evm.tar.gz -C
/media/LABEL2
sync
```

```
umount /media/LABEL1  
umount /media/LABEL2
```

3. After the above commands are executed, power on the board and then hit any key on your keyboard when you see the prompt "Hit any key to stop autoboot:", to enter u-boot mode as shown below:

```
CCCCCCCC  
U-Boot SPL 2011.09-svn (May 03 2012 - 10:49:04)  
Texas Instruments Revision detection unimplemented  
Booting from MMC...  
OMAP SD/MMC: 0  
reading u-boot.img  
reading u-boot.img  
  
U-Boot 2011.09-svn (May 03 2012 - 10:49:04)  
  
I2C:  ready  
DRAM:  512 MiB  
WARNING: Caches not enabled  
Did not find a recognized configuration, assuming General purpose EVM in Profile  
0 with Daughter board  
NAND:  HW ECC Hamming Code selected  
512 MiB  
MMC:  OMAP SD/MMC: 0  
*** Warning - bad CRC, using default environment  
  
Net:  cpsw  
Hit any key to stop autoboot:  0 (input any key here)
```

4. Enter the following commands. The system will continue the booting process.

```
Devkit8600# setenv bootargs console=ttyO0,115200n8 earlyprintk  
root=/dev/mmcblk0p2 rw rootfstype=ext3 rootwait  
Devkit8600# setenv bootcmd 'mmc rescan;fatload mmc 0 80300000 uImage;bootm  
80300000'  
Devkit8600# saveenv  
Saving Environment to NAND...  
Erasing Nand...  
Erasing at 0x260000 -- 100% complete.  
Writing to Nand... done  
Devkit8600# boot
```

```
reading uImage

2949384 bytes read
## Booting kernel from Legacy Image at 80300000 ...
   Image Name:   Linux-3.1.0
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    2949320 Bytes = 2.8 MiB
   Load Address: 80008000
   Entry Point:  80008000
   Verifying Checksum ... OK
   Loading Kernel Image ... OK
OK

Starting kernel ...

.....
Arago Project http://arago-project.org am335x-evm ttyO0

Arago 2011.09 am335x-evm ttyO0

am335x-evm login: root
```

5. The TISDK file system has some preinstalled application programs, which are based on the QT UI; they can be executed by the user easily.

5.10 Application Development

This section mainly introduces the development of an application program, and illustrates the general process of application development with an example.

5.10.1 Development example: LED control program

1. To Edit code

led_acc.c source code: causes LED lamps on the development board to flicker.

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/ioctl.h>
#include <fcntl.h>
```

```
#define LED1 "/sys/class/leds/sys_led/brightness"
#define LED2 "/sys/class/leds/user_led/brightness"

int main(int argc, char *argv[])
{
    int f_led1, f_led2;
    unsigned char i = 0;
    unsigned char dat1, dat2;
    if((f_led1 = open(LED1, O_RDWR)) < 0){
        printf("error in open %s",LED1);
        return -1;
    }
    if((f_led2 = open(LED2, O_RDWR)) < 0){
        printf("error in open %s",LED2);
        return -1;
    }

    for(;;){
        i++;
        dat1 = i&0x1 ? '1':'0';
        dat2 = (i&0x2)>>1 ? '1':'0';
        write(f_led1, &dat1, sizeof(dat1));
        write(f_led2, &dat2, sizeof(dat2));
        usleep(300000);
    }
}
```

2. To Cross-compile

```
arm-none-linux-gnueabi-gcc led_acc.c -o led_acc
```

3. Download and run

Upload to the development board system via TF card, USB flash disk or the network, enter the directory with the led_acc file, input the following commands and press Enter, to run led_acc in the background.

```
./led_acc &
```

6 Windows CE 7 Operating System

6.1 Introduction

This section mainly introduces the Devkit8600 system and application development for Windows Embedded Compact 7, as well as software resources on the included CD, software features, establishment of development environment, and how to sysgen and build a BSP (board support package) and so on.

6.1.1 Software Resources

- BSP (Board Support Package)
- CD\WINCE700\BSP\AM33x_BSP.rar
- CD\WINCE700\BSP\COMMON_TI_V1.rar
- CD\WINCE700\BSP\3rdParty.rar
- CD\WINCE700\BSP\PowerVR.rar

Windows Embedded Compact 7 sample project

- CD\WINCE700\project\AM335X_OS

Sample application

- CD\WINCE700\app\

Pre-compiled image

- CD\WINCE700\Image\

MLO	First bootloader for TF card boot
xldrnd.nb0	First bootloader for NAND flash boot
Ebootsd.nb0	Second bootloader for TF card boot
Ebootnd.nb0	Second bootloader for NAND flash boot

Nk.bin

WinCE runtime image

6.2 Software Features

Contents of BSP:

Catalogue	Item	Source code / binary
X-Loader (First boot loader)	NAND	Source
	SD	Source
EBOOT (Second boot loader)	NAND	Source
	SD	source
OAL	Boot parameter	Source
	KILT(EMAC)	Source
	Serial debug	Source
	REBOOT	Source
	Watchdog	Source
	RTC	Source
	Kernel profiler	Source
	System timer	Source
	Interrupt controller	Source
	MMU	Source
Driver	NLED driver	Source
	GPIO/I2C/SPI/MCASP driver	Source
	Serial port driver	Source
	Audio driver	Source
	NAND driver	Source
	Display driver	Source

	TOUCH driver	Source
	SD/MMC/SDIO driver	Source
	EMAC driver	Source
	USB OTG driver	Source
	GPIO keyboard driver	Source
	DMA driver	Source
	Backlight driver	Source
	Battery driver	Source
	RPU driver	Source
SDK	powerVR DDK & SDK	Binary & Source

6.3 System Development

6.3.1 Installation of Integrated Development Environment

Please install the following items (in order) to windows XP:

- 1) Visual Studio 2008
- 2) Visual Studio 2008 SP1
- 3) Windows Embedded Compact 7
- 4) Windows Embedded Compact 7 Updates
- 5) ActiveSync 4.5

Note:

 The CD does not provide Windows Embedded Compact 7 development environment tools, please download them from:



<http://www.microsoft.com/download/en/default.aspx>

6.3.2 Extract BSP and project files to IDE

The following preparations should be made:

1. Extract [CD\WINCE700\BSP\AM33x_BSP.rar] to [C:\WINCE700\PLATFORM] directory.
2. Extract [CD\WINCE700\BSP\COMMON_TI_V1.rar] to [C:\WINCE700\PLATFORM\COMMON\SRC\SOC].
3. Extract [CD\WINCE700\BSP\3rdParty.rar] to [C:\WINCE700].
4. Extract [CD\WINCE700\BSP\powerVR.rar] to [C:\WINCE700\public].
5. Copy directory [CD\WINCE700\project\AM335X_OS] to [C:\WINCE700\OSDesigns] directory.

Note:

 The default installation path of the Windows Embedded Compact 7 is [C:\WINCE700]. It is assumed for this tutorial that your installation path is identical

6.3.3 Sysgen & Build BSP

Below are the steps given for Sysgen and BSP building:

- 1) Open the existing project file AM335X_OS.sln located in [C:\WINCE700\OSDesigns\AM335X_OS].
- 2) Click **Build** then **Build Solution** in vs2008 to sysgen and build the BSP.
- 3) Images including MLO, EBOOTSD.nb0, NK.bin will be created after the sysgen phase and the build phase are finished successfully, Copy the files MLO, EBOOTSD.nb0, and NK.bin from:
[C:\WINCE700\OSDesigns\AM335X_OS\AM335X_OS\RelDir\AM33X_BSP_ARMV7_Release] to the TF card.
- 4) Insert the TF card into the device and boot the device for a test.

6.3.4 Driver Introduction

Source code path of all drivers in BSP:

Driver	Source Code Location
NLED driver	BSP\AM33X_BSP\SRC\DRIVERS\NLED
GPIO	BSP\AM33X_BSP\SRC\DRIVERS\GPIO BSP\COMMON_TI_V1\COMMON_TI_AMXX\GPIO
I2C	BSP\COMMON_TI_V1\COMMON_TI_AMXX\OAL\OALI2C
SPI	BSP\COMMON_TI_V1\COMMON_TI_AMXX\SPI BSP\AM33X_BSP\SRC\DRIVERS\MCSPI
MCASP driver	BSP\COMMON_TI_V1\COMMON_TI_AMXX\MCASP
Serial port driver	BSP\COMMON_TI_V1\COMMON_TI_AMXX\SERIAL BSP\AM33X_BSP\SRC\DRIVERS\UART
Audio driver	BSP\AM33X_BSP\SRC\DRIVERS\WAVEDEV2
NAND driver	BSP\AM33X_BSP\SRC\DRIVERS\BLOCK BSP\COMMON_TI_V1\COMMON_TI_AMXX\BLOCK
Display driver	BSP\COMMON_TI_V1\COMMON_TI_AMXX\DSS_Netra BSP\AM33X_BSP\SRC\DRIVERS\DISPLAY
TOUCH driver	BSP\AM33X_BSP\SRC\DRIVERS\TOUCH
SD/MMC/SDIO driver	BSP\AM33X_BSP\SRC\DRIVERS\SDHC BSP\COMMON_TI_V1\COMMON_TI_AMXX\SDHC BSP\COMMON_TI_V1\COMMON_TI\SDHC
EMAC driver	BSP\COMMON_TI_V1\AM33X\CPSW3Gminiport BSP\AM33X_BSP\SRC\DRIVERS\EMAC
USB OTG driver	BSP\AM33X_BSP\SRC\DRIVERS\USB BSP\COMMON_TI_V1\AM33X\USB
GPIO keyboard driver	BSP\AM33X_BSP\SRC\DRIVERS\KEYPAD
Backlight driver	BSP\AM33X_BSP\SRC\DRIVERS\BACKLIGHT
Battery driver	BSP\ AM33X_BSP \SRC\DRIVERS\BATTERY

Driver	Source Code Location
PRU driver	BSP\COMMON_TI_V1\AM33X\PRU BSP\ AM33X_BSP\SRC\DRIVERS\PRU
DMA driver	BSP\ AM33X_BSP\SRC\DRIVERS\EDMA BSP\COMMON_TI_V1\COMMON_TI_AMXX\EDMA

If you wish to refer to more Windows Embedded Compact 7 driver development material, the specific reference document of PB7.0 is available from:

- 🖱 Start
- 🖱 All programs
- 🖱 Microsoft Visual Studio 2008
- 🖱 Microsoft Visual Studio 2008 Document
- 🖱 Content(C)
- 🖱 Windows Embedded Compact 7
- 🖱 Device Driver.

6.4 Updating System Image

The Devkit8600 supports boot-up from TF card and NAND; this section will introduce the two different system update methods.

6.4.1 Updating the TF card image

Format TF card

HP USB Disk Storage Format Tool 2.0.6 is recommended:

The software can be downloaded from



<http://www.embest-tech.com/resource/download/HP-USB-Disk-Storage-Format-Tool.rar>.

- 1) Insert TF card into the card reader in the PC.

- 2) Open the HP USB Disk Storage Format Tool, the following steps will be shown in detail.

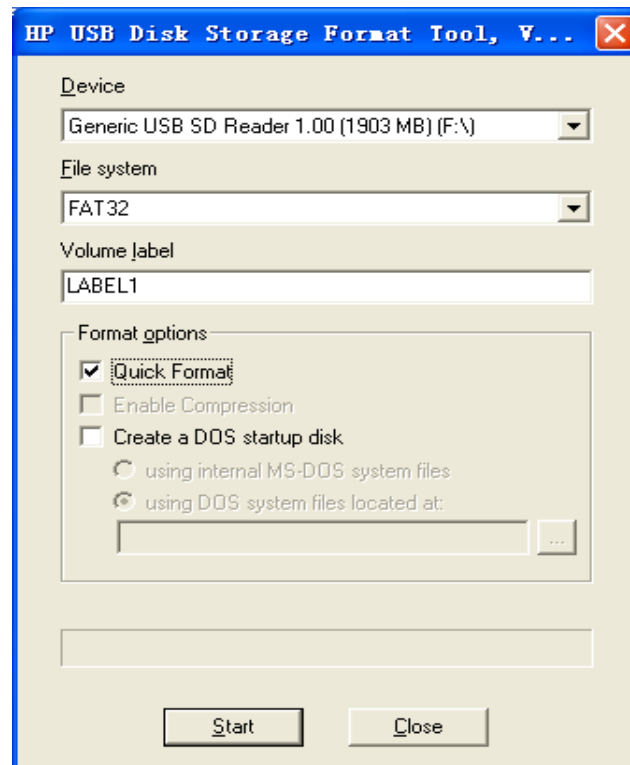


Figure 30: USB Disk Storage Format Tool

1. Select "FAT32".
2. Click "Start".
3. When formatting is completed, click "OK".

Note:

-  HP USB Disk Storage Format Tool will clear partitions of the TF card.
-  Use other format tool may cause the failure of the TF card booting

Copy runtime image

Copy the **MLO**, **EBOOTSD.nb0** and **NK.bin** image files located in CD\WINCE700\image to the TF card.

Boot system

Insert a TF card and reboot the system. And then, the system will boot from the TF card. Press the space button and enter EBOOT menu. To select boot device and LCD module display output, follow these steps:

1) Enter EBOOT menu

```
CCCCCCCC
Texas Instruments Windows CE SD X-Loader33X
Built Jul 27 2012 at 11:25:59
Version BSP_WINCE_ARM_A8 02.30.00.03
open ebootsd.nb0 file
Init HW: controller RST
SDCARD: requested speed 1000000, actual speed 1000000
SDCARD: requested speed 25000000, actual speed 19200000
read ebootsd.nb0 file

jumping to ebootsd image

Microsoft Windows CE Bootloader Common Library Version 1.4 Built Jul 27 2012
11:23:05
I2C EEPROM returned wrong magic value 0xffffffff
INFO:OALLogSetZones: dpCurSettings.ulZoneMask: 0x8409

Texas Instruments Windows CE EBOOT for AM33x, Built Jul 27 2012 at 11:25:53
EBOOT Version 0.0.1, BSP BSP_WINCE_ARM_A8 02.30.00.03
AHCLKX pinmux:0
AHCLKX CTRL:0x8001
pin function:0x0
pin dir:0x8000000

TI AM33X

ecc type:3
System ready!
Preparing for download...
INFO: Predownload....
Checking bootloader blocks are marked as reserved (Num = 18)

BOOT_CFG_SIGNATURE is different, read -1, expect 1111705159
WARN: Boot config wasn't found, using defaults
INFO: SW3 boot setting: 0x04
IsValidMBR: MBR sector = 0x480 (valid MBR)
OpenPartition: Partition Exists=0x1 for part 0x20.
```

```
>>> Forcing cold boot (non-persistent registry and other data will be wiped) <<<
e0311800 56e4 -> 0 18 31 e0 e4 56
e0311800 57e4 -> 0 18 31 e0 e4 57
Hit space to enter configuration menu [56] 5... (Press the space key to enter EBOOT
menu)
```

4. Type [2]->[2] select boot from TF card

```
-----
----
Main Menu
-----
----
[1] Show Current Settings
[2] Select Boot Device
[3] Select KITL (Debug) Device
[4] Network Settings
[5] SDCard Settings
[6] Set Device ID
[7] Save Settings
[8] Flash Management
[9] Enable/Disable OAL Retail Messages
[a] Select Display Resolution
[b] Select OPP Mode
[0] Exit and Continue

Selection: 2

-----
----
Select Boot Device
-----
----
[1] Internal EMAC
[2] NK from SDCard FILE
[3] NK from NAND
[0] Exit and Continue

Selection (actual Internal EMAC): 2
Boot device set to NK from SDCard FILE
```

5. Type [a] to select the LCD/LVDS module display output

```
-----
----
```

Main Menu

- ```

[1] Show Current Settings
[2] Select Boot Device
[3] Select KITL (Debug) Device
[4] Network Settings
[5] SDCard Settings
[6] Set Device ID
[7] Save Settings
[8] Flash Management
[9] Enable/Disable OAL Retail Messages
[a] Select Display Resolution
[b] Select OPP Mode
[0] Exit and Continue
```

Selection: **a**

```


Select Display Resolution

```

- ```
-----  
[1] LCD 480x272 60Hz //For 4.3-inch LCD  
[2] DVI 640x480 60Hz (N/A)  
[3] DVI 640x480 72Hz (N/A)  
[4] LCD 800x480 60Hz //For 7-inch LCD  
[5] DVI 800x600 60Hz (N/A) //For LVDS  
[6] DVI 800x600 56Hz (N/A)  
[7] VGA 1024x768 60Hz //For VGA  
[8] DVI 1280x720 60Hz (N/A)  
[0] Exit and Continue Selection (actual LCD 480x272 60Hz): 4
```

6. Type [0] to boot the system

```
-----  
-----  
Main Menu  
-----
```

- ```

[1] Show Current Settings
[2] Select Boot Device
[3] Select KITL (Debug) Device
[4] Network Settings
[5] SDCard Settings
```

```

[6] Set Device ID
[7] Save Settings
[8] Flash Management
[9] Enable/Disable OAL Retail Messages
[a] Select Display Resolution
[b] Select OPP Mode
[0] Exit and Continue

Selection: 0
mode = 3
LcdPdd_LCD_GetMode:3
mode = 3
LcdPdd_LCD_Initialize:3
OEMPreDownload: Filename nk.bin
Init HW: controller RST
SDCARD: requested speed 1000000, actual speed 1000000
SDCARD: requested speed 25000000, actual speed 19200000

BL_IMAGE_TYPE_BIN

+OEMMultiBinNotify(0x8feb24d8 -> 1)
Download file information:

[0]: Address=0x80002000 Length=0x03c9e9bc Save=0x80002000

Download file type: 1
+OEMIsFlashAddr(0x80002000) g_eboot.type 1
.....
.....
.....
.....rom_offset=0x0.
..ImageStart = 0x80002000, ImageLength = 0x3c9e9bc, LaunchAddr = 0x8000b6a0

Completed file(s):

+OEMIsFlashAddr(0x80002000) g_eboot.type 1
[0]: Address=0x80002000 Length=0x3c9e9bc Name="" Target=RAM
ROMHDR at Address 80002044h
Launch Windows CE image by jumping to 0x8000b6a0...

Windows CE Kernel for ARM (Thumb Enabled)
CPU CP15 Control Register = 0xc5387f

```

```
CPU CP15 Auxiliary Control Register = 0x42
I2C EEPROM returned wrong magic value 0xffffffff
+OALTimerInit(1, 24000, 200)
--- High Performance Frequency is 24 MHz---
```

## 6.4.2 Update NAND Flash image

Format TF card

Please refer to p85.

Copy runtime image

Copy MLO, EBOOTND.nb0, NK.bin, XLDRNAND.nb0 and EBOOTSD.nb0 image files located in CD\WINCE700\image to the TF card.

flashing image file

Insert the TF card and reboot the system. The system will then boot from the TF card. The HyperTerminal will display a boot message, you can press [SPACE] to enter the EBOOT menu. An image can be flashed to NAND flash according to the following steps:

- 1) Press [8] to enter the Flash menu.
- 2) Press[9]->[4]->[A], [9]->[3]->[B] and[9]->[2]->[C] to write XLDR, EBOOT and NK images respectively.
- 3) Then press [0] to return to main menu, and press [2], [3] Select boot from NAND flash, then type [A] select the LCD,VGA or LVDS output mode [7] and [y] to save the Boot setting.

Unplug the TF card and then reboot the system, and then the system will boot from NAND Flash.

## 6.5 Instructions for use

### 6.5.1 How to use openGL ES demo

- 1) Select PowerVR items in catalog items view in VS2008 as below:

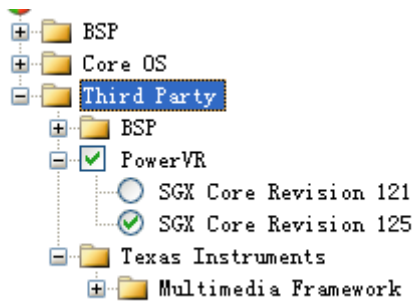


Figure 4-1

1. Click [Build-> Build Solution] in VS2008 menu, after sysgen and build BSP have finished, replace the nk.bin located in the TF card with the newly generated nk.bin
2. Copy  
C:\WINCE700\PUBLIC\PowerVR\oak\target\Rev125\ARMV4I\retail\\*.exe to the Devkit8600 windows embedded compact 7 system.  
And double click each demo to test.

## 6.6 Application Development

This chapter introduces how to develop Windows Embedded Compact 7 application programs in Devkit8600.

### 6.6.1 Application program interfaces and examples

The API used for development of Devkit8600 application programs employs the Microsoft Windows Embedded Compact 7 standard application program interface definition, Devkit8600 only expands interface definition of GPIO based on standard API. Please refer to the CD\WINCE700\app\GPIOAppDemo to see how to control the GPIO pin status.

Please check relative Help documents of MSDN Windows Embedded Compact 7 API for Windows Embedded Compact 7 standard application program interface definition.

### 6.6.2 GPIO application program interfaces and examples

GPIO device name is L"GPIO1:" The extension the DeviceIoControl interface definition corresponding device IOCTL code includes:

| IOCTL Code         | Description                            |
|--------------------|----------------------------------------|
| IOCTL_GPIO_SETBIT  | Set GPIO pin as 1                      |
| IOCTL_GPIO_CLRBIT  | Set GPIO pin as 0                      |
| IOCTL_GPIO_GETBIT  | Read GPIO pin                          |
| IOCTL_GPIO_SETMODE | Set the working mode of GPIO pin       |
| IOCTL_GPIO_GETMODE | Read the working mode of GPIO pin      |
| IOCTL_GPIO_GETIRQ  | Read the corresponding IRQ of GPIO pin |

An example is shown below:

### 1) Open the GPIO device

```
HANDLE hFile = CreateFile (_T ("GPIO1:"), (GENERIC_READ|GENERIC_WRITE),
(FILE_SHARE_READ|FILE_SHARE_WRITE), 0, OPEN_EXISTING, 0, 0);
```

### 1. Set the working mode of the GPIO

```
DWORD id = 48, mode = GPIO_DIR_OUTPUT;
```

Set the working mode of the GPIO:

```
DWORD pInBuffer [2];
pInBuffer [0] = id;
pInBuffer [1] = mode;
DeviceIoControl (hFile, IOCTL_GPIO_SETMODE, pInBuffer, sizeof (pInBuffer), NULL,
0, NULL, NULL);
```

Read the working mode of the GPIO:

```
DeviceIoControl (hFile, IOCTL_GPIO_GETMODE, &id, sizeof(DWORD), &mode,
sizeof(DWORD), NULL, NULL);
```

"id" is the GPIO Pin number, "mode" is the GPIO mode, including:

| Mode definition | Description |
|-----------------|-------------|
| GPIO_DIR_OUTPUT | Output mode |
| GPIO_DIR_INPUT  | Input mode  |

|                      |                           |
|----------------------|---------------------------|
| GPIO_INT_LOW_HIGH    | Rising edge trigger mode  |
| GPIO_INT_HIGH_LOW    | Falling edge trigger mode |
| GPIO_INT_LOW         | low level trigger mode    |
| GPIO_INT_HIGH        | high level trigger mode   |
| GPIO_DEBOUNCE_ENABLE | Jumping trigger enable    |

### 1) The operation of GPIO Pin:

```
DWORD id = 48, pinState = 0;
```

#### Output high level:

```
DeviceIoControl (hFile, IOCTL_GPIO_SETBIT, &id, sizeof (DWORD), NULL, 0,
NULL, NULL);
```

#### Output low level:

```
DeviceIoControl (hFile, IOCTL_GPIO_CLRBIT, &id, sizeof (DWORD), NULL, 0,
NULL, NULL);
```

#### Read the pin state

```
DeviceIoControl (hFile, IOCTL_GPIO_GETBIT, &id, sizeof (DWORD), &pinState,
sizeof (DWORD), NULL, NULL);
```

"id" is the GPIO pin number, "pinState" returns to pin state

## 2. Other optional operations

#### Read the corresponding IRQ number of a GPIO pin

```
DWORD id = 0, irq = 0;
DeviceIoControl (hFile, IOCTL_GPIO_GETIRQ, &id, sizeof (DWORD), &irq, sizeof
(DWORD), NULL, NULL);
```

"id" is the GPIO pin number, "irq" returns the IRQ number

## 3. Close the GPIO device

```
CloseHandle (hFile);
```

#### Note:

 Definition of GPIO pin: 0~127 MPU Bank0~3 GPIO pin.

 GPIO pin 0~127 has to be configured as a GPIO under AM33X\_BSP/SRC/inc/bsp\_padcfg.h when referred to as a GPIO pin.

# Appendix 1: Installing an Ubuntu Linux System

An appropriate development environment is required for software development. The CD included with the product contains a development environment which needs to be installed under a Linux environment. If you are working on a PC running Windows, you have to create a Linux system first, and then you can install the environment. An easy method for achieving this is to use virtual machine software such as VirtualBox to install Ubuntu Linux on an emulated/virtual PC. The following sections will introduce the installation processes of VirtualBox and an Ubuntu system.

## 1.1 Installing VirtualBox

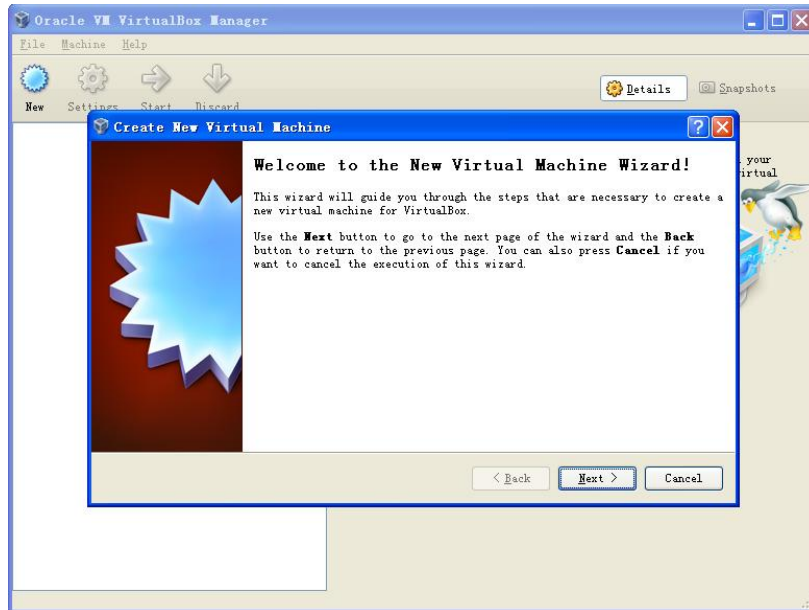
The latest version of virtual box can be downloaded from:



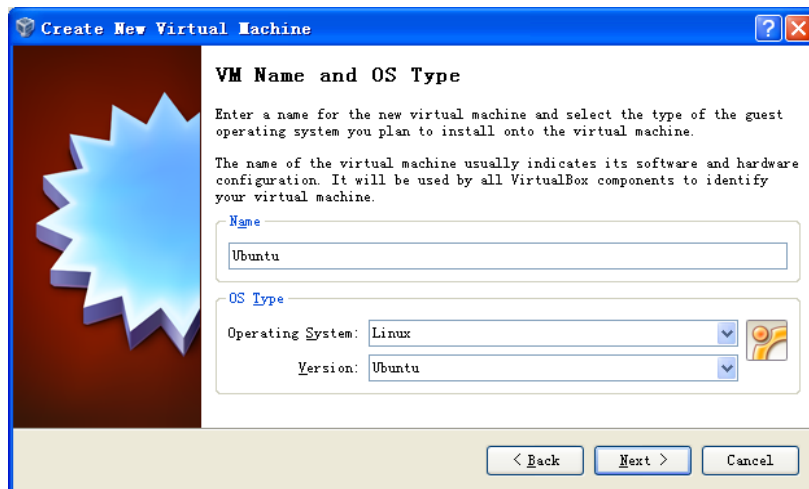
<http://www.virtualbox.org/wiki/Downloads>

VirtualBox requires a minimum of 512MB of RAM to run however 1GB is recommended. Installation is simple and instructions have been provided below:

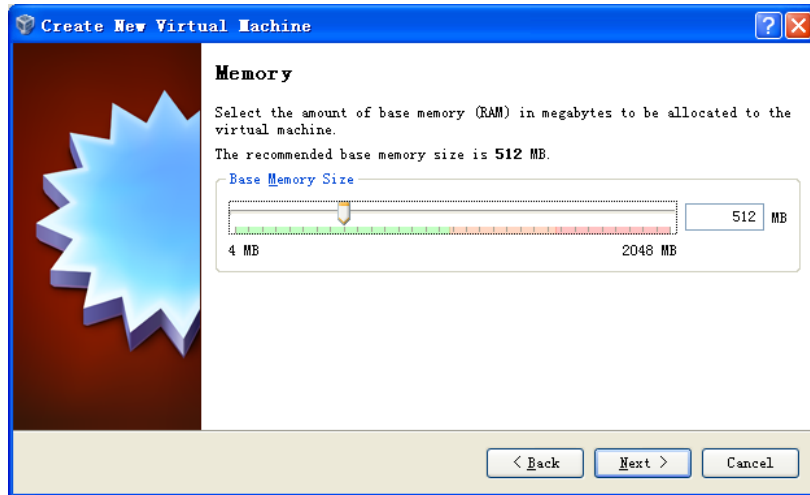
1. Start VirtualBox from the Start menu of Windows, and then click New in the VirtualBox window. A pop-up window titled "Create New Virtual Machine" will be shown as below:



2. Click Next to create a new virtual machine.
3. Enter a name for the new virtual machine and select the operating system type as shown below:



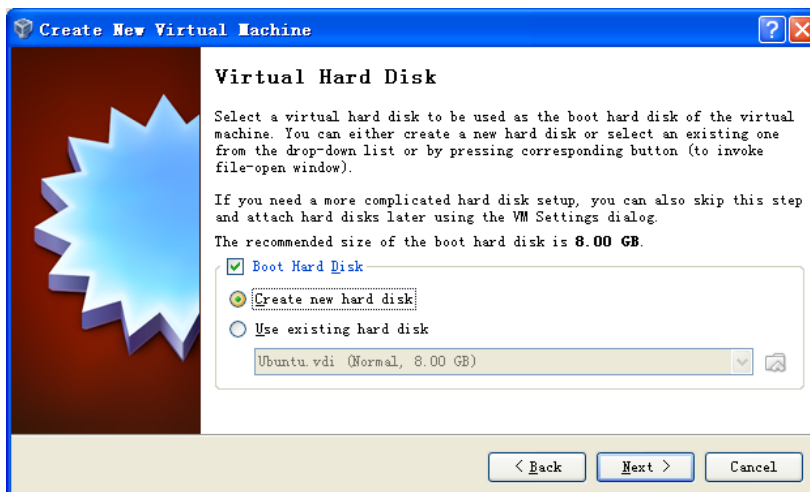
4. Enter a name in the Name field, e.g. Ubuntu, and select Linux in the Operating System drop-down menu, and then click Next.
5. Allocate memory to the virtual machine and then click Next



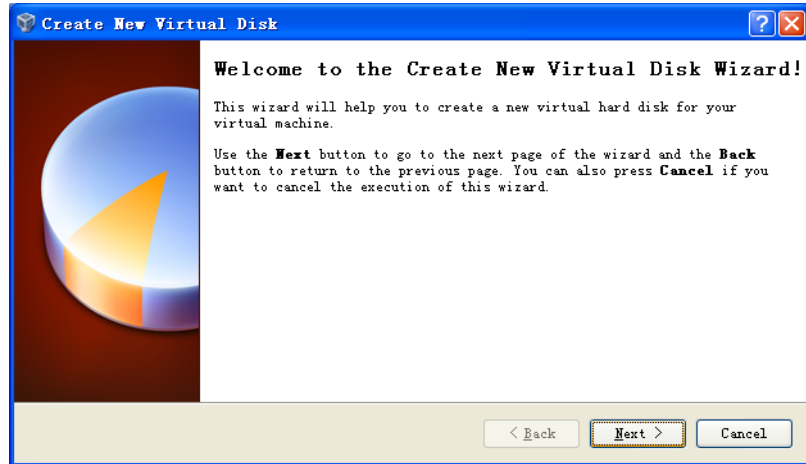
**Note:**

-  If your PC has 1GB of RAM or lower, keep the default setting;
-  If your PC has more than 1GB of RAM, you can allocate up to 1/4 to the virtual machine, for example, 512MB out of 2GB memory could be allocated to virtual machine.

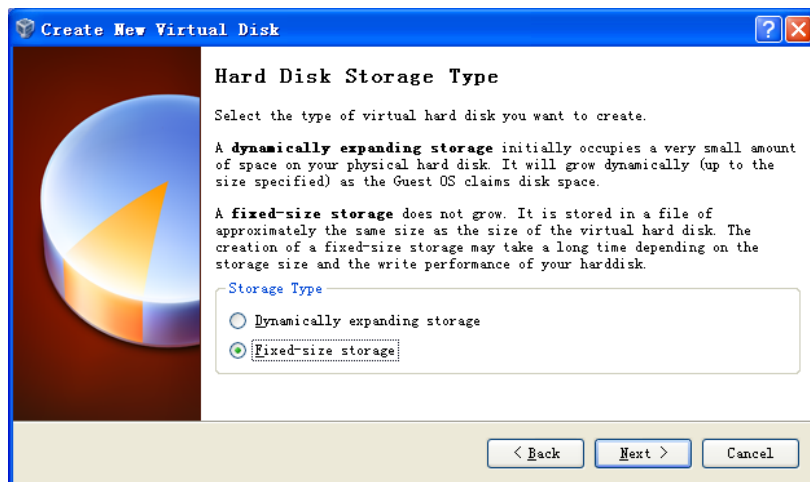
6. If this is the first time you have installed VirtualBox, you need to select Create new hard disk in the following window, and then click Next



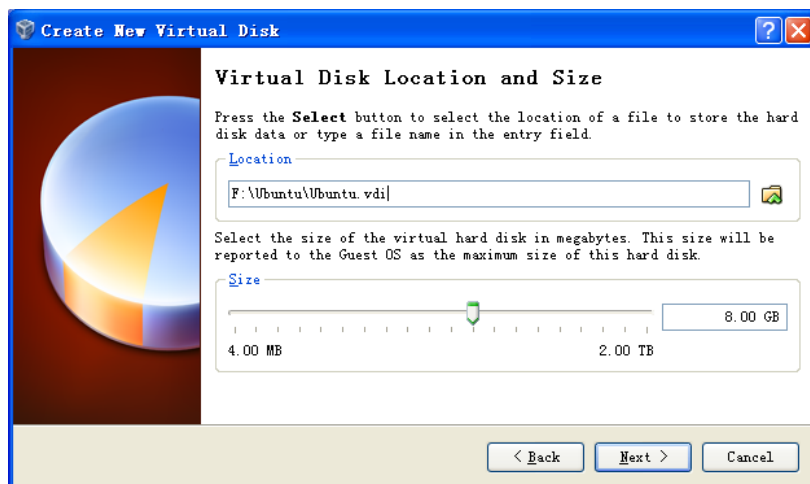
7. Click Next in the following window



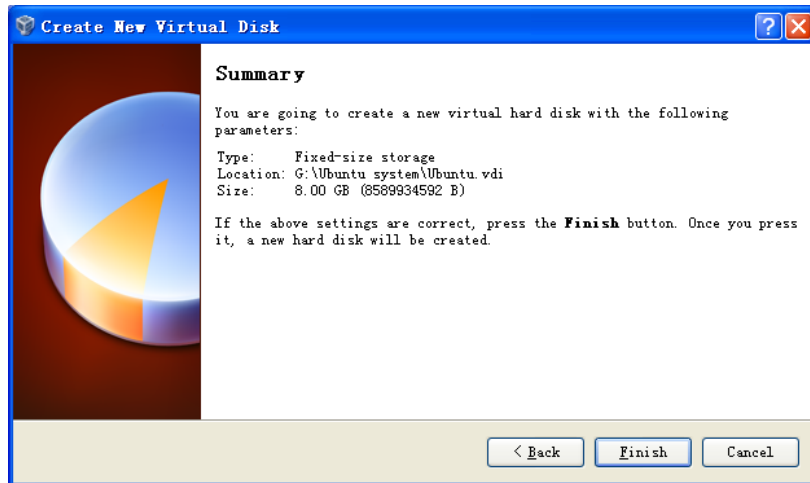
8. Select Fixed-size storage in the following window and click Next



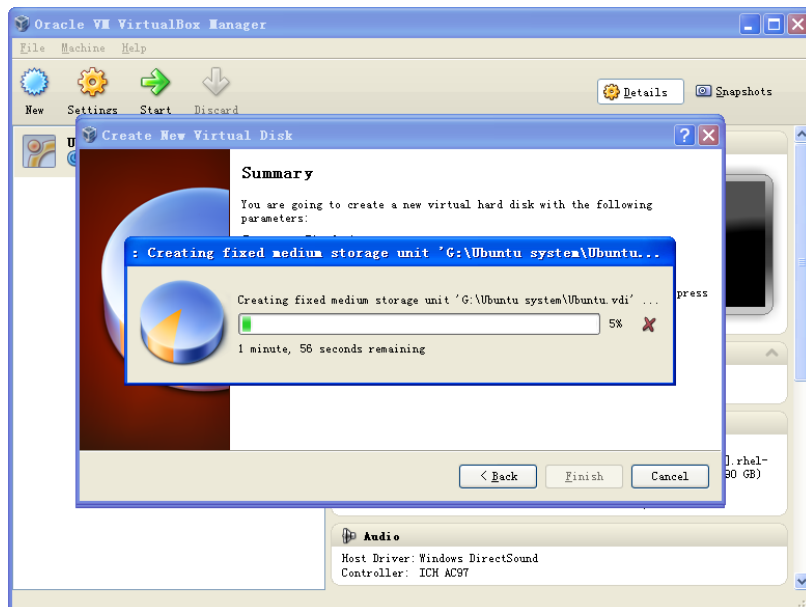
9. Define where the hard disk data is stored and the default space of the virtual disk (8GB at least), and then click Next



10. Click Finish in the following window



11. Your PC will then create a new virtual disk



12. A window with summary of the newly created virtual machine will be shown as below when the creation process is done. Please click Finish to complete the whole process.



## 1.2 Installing the Ubuntu Linux System

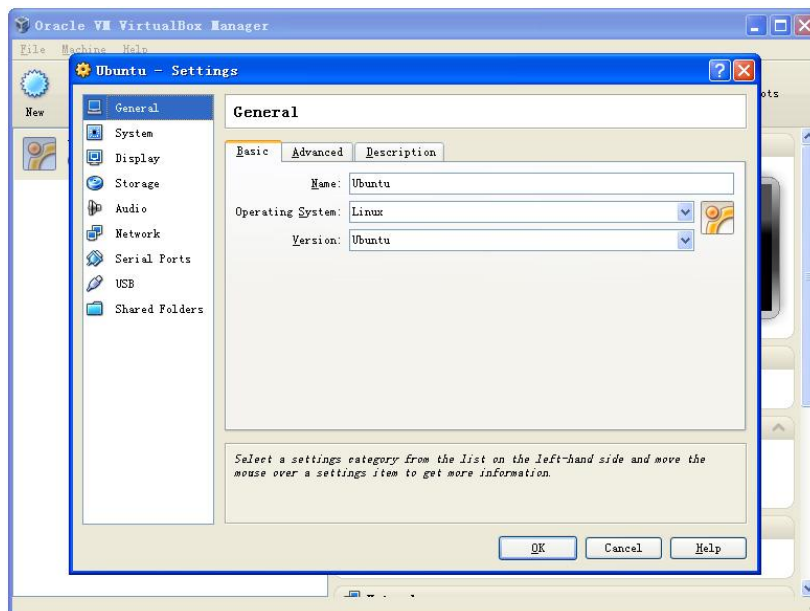
After VirtualBox is installed, we can install the Ubuntu Linux system. Visit:



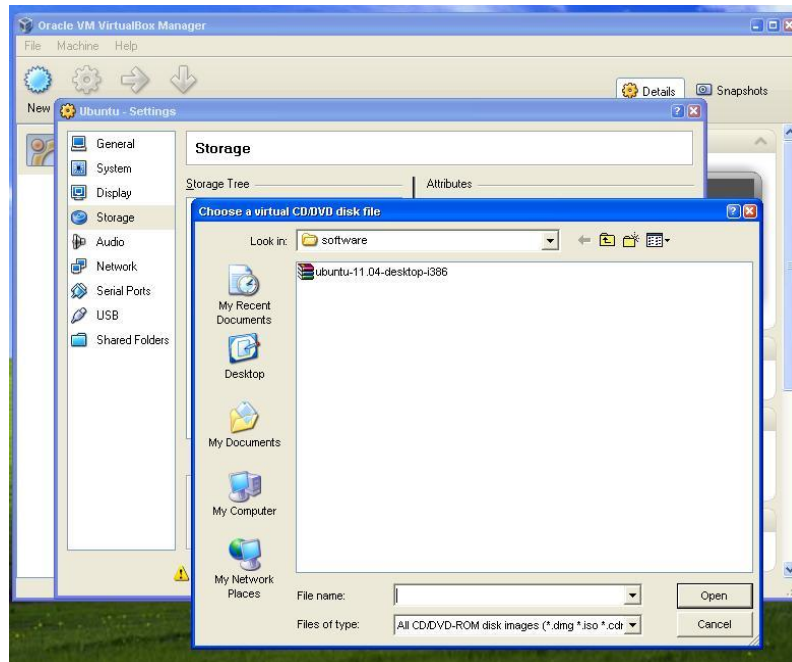
<http://www.Ubuntu.com/download/Ubuntu/download>

to download the ISO image file of Ubuntu and then follow the steps below:

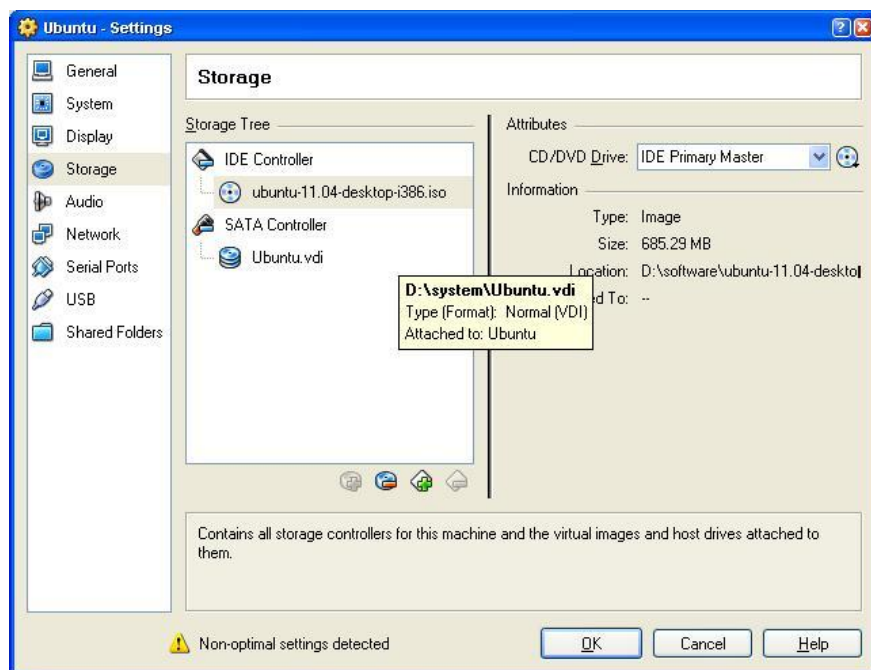
1. Start VirtualBox from the Start menu and click Settings on the VirtualBox window. A Settings window will be shown as below



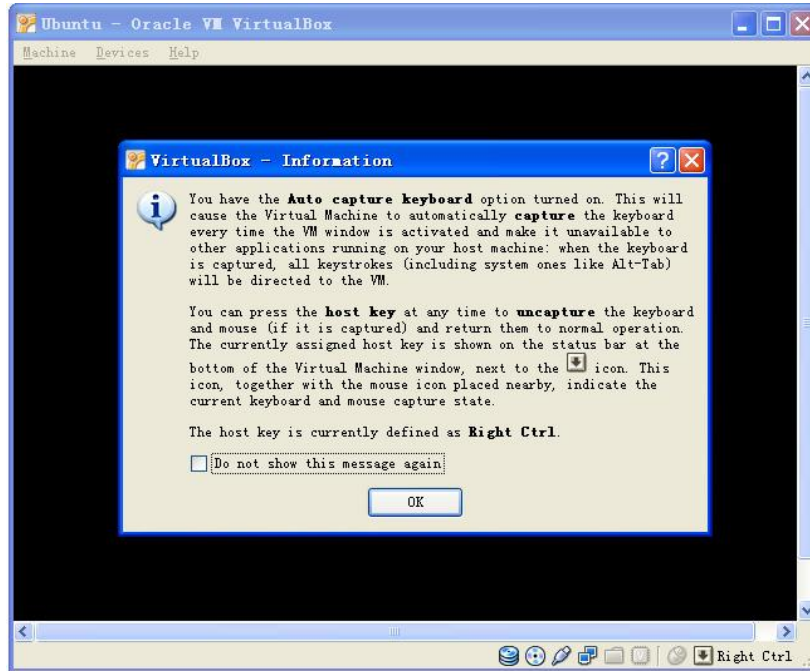
2. Select Storage on the left side of the Settings window and click the CD icon next to the option Empty under IDC controller to the right side of the window, and then find the ISO file you downloaded



3. Select the ISO file you downloaded and click OK as shown below



4. Click Start on the VirtualBox window, the Ubuntu installation program will start as shown below:



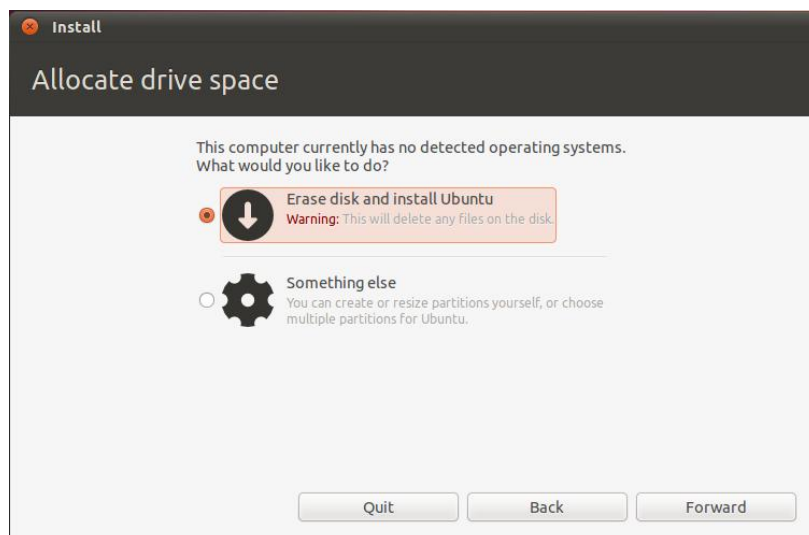
5. Some prompt windows will pop up during the initiation process. You need only click OK all the way to the end of the process.
6. Click Install Ubuntu to start installation when the following window appears



7. Click Forward to continue the process



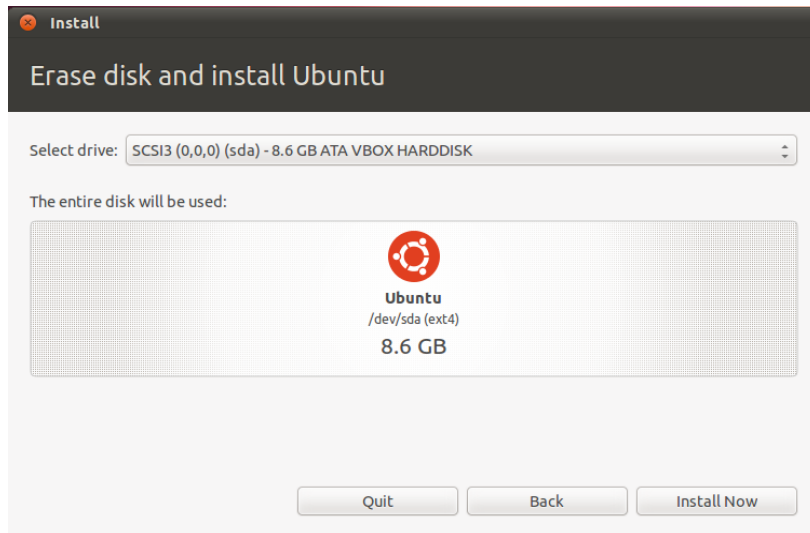
8. Select **Erase disk and install Ubuntu** and click **Forward**



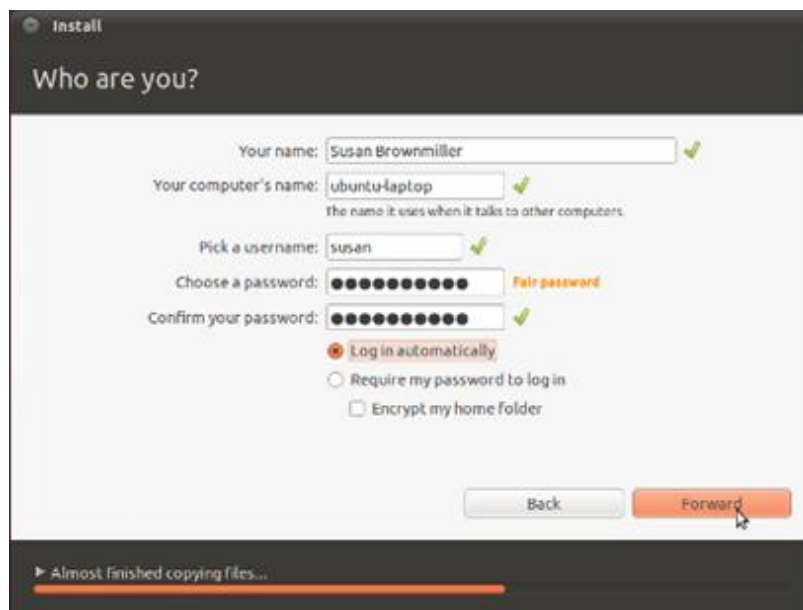
**Note:**

 Selecting this option will only affect the virtual hard drive you created earlier and will not lead to any content loss on your physical hard drive.

9. Click Install Now in the following window to start the installation:



10. Some simple questions need to be answered during the installation process. Please enter appropriate information and click Forward. The following window is the last question that will appear during the process:

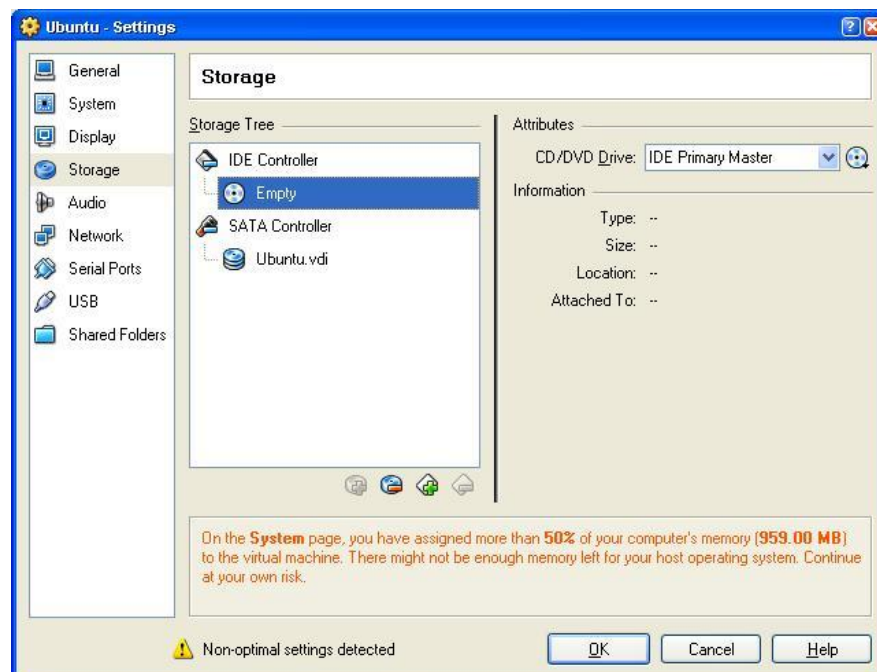


11. After all the required information is properly entered in to the fields, select **Log in automatically** and click **Forward**.
12. The installation of Ubuntu may take between 15 minutes to an hour depending on your PC. A prompt window will be shown as below after installation is done. Please select **Restart Now** to restart the Ubuntu system.



### Note:

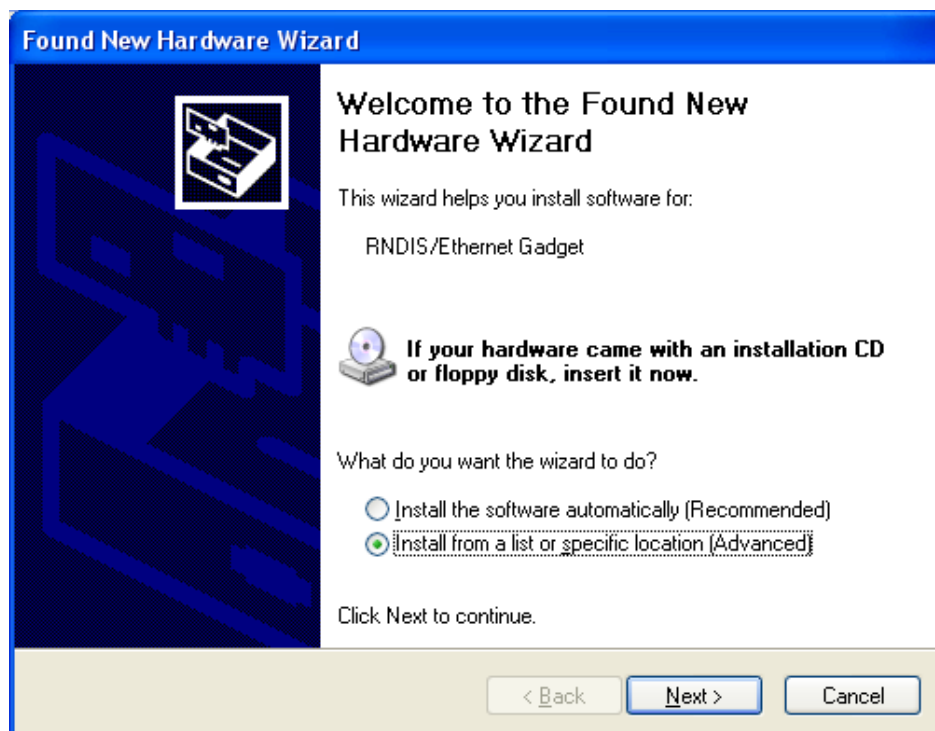
 The Normally the ISO file shown below will be ejected automatically by VirtualBox after restarting Ubuntu. If it is not, you can eject the ISO file manually in the **Settings** window of VirtualBox. The following window shows the settings window after the ISO file is ejected.



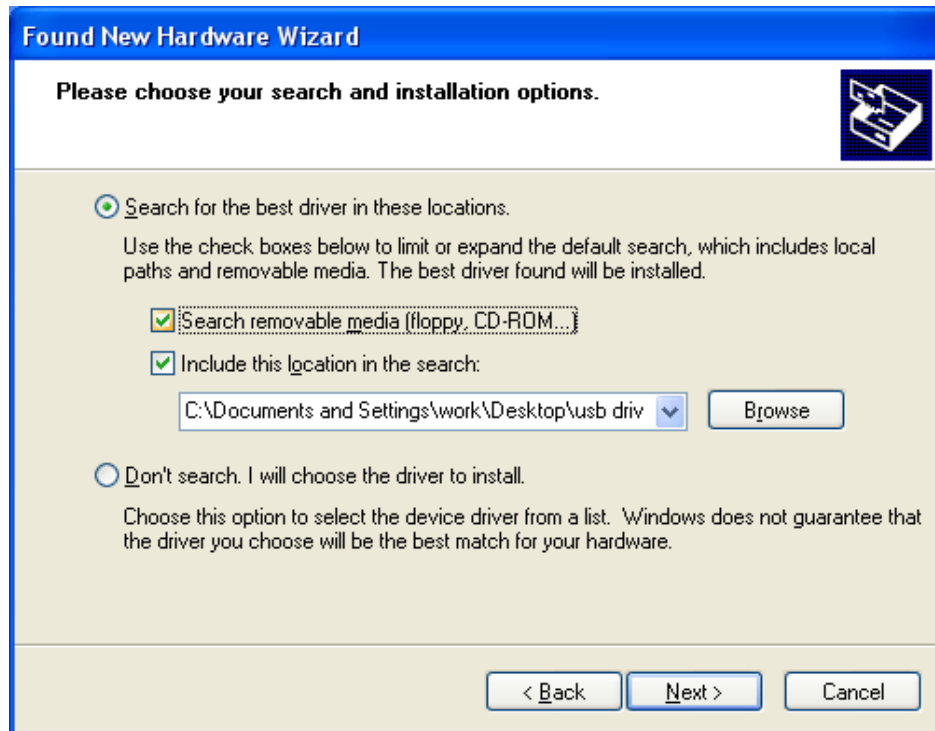
13. Once the restart is complete, the Ubuntu system is ready for use.

## Appendix 2: Driver Installation Of Linux USB Ethernet/RNDIS Gadget

1. If you don't install drivers for the Linux USB Ethernet/RNDIS Gadget, the PC will find the new hardware and give you a hint on the screen, please select "From list or designated location", then click "Next"



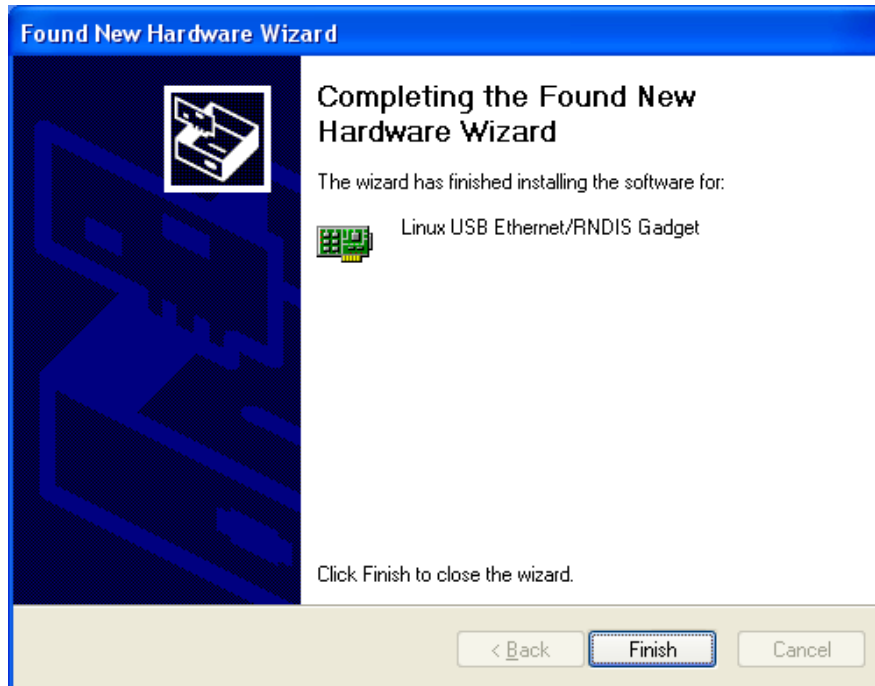
2. Designate the path for the usb driver, the usb driver directory is [disk\linux\tools], then click "Next"



3. When the following appears, select "Continue"



4. Please wait until the installation is completed



# Appendix 3: Linux Boot Disk Format

Instructions for creation of a dual-partition card to allow the DevKit8500D/A to boot Linux from the first partition and have root file system in the second partition.

## 6.1 Introduction

This guide is meant for those looking to create a dual-partition card, booting from a FAT partition that can be read by the OMAP3 ROM bootloader and Linux/Windows, and then utilizing an ext3 partition for the Linux root file system.

### Note:

 Text marked with [] shows user input.

### 6.1.1 Determine the location of the TF Card Reader on your system

Plug the TF Card into the TF Card Reader and then plug the TF Card Reader into your system. The following procedure can be used to determine which device it is on your system.

```
$ [dmesg | tail]
...
[6854.215650] sd 7:0:0:0: [sdc] Mode Sense: 0b 00 00 08
[6854.215653] sd 7:0:0:0: [sdc] Assuming drive cache: write through
[6854.215659] sdc: sdc1
[6854.218079] sd 7:0:0:0: [sdc] Attached SCSI removable disk
[6854.218135] sd 7:0:0:0: Attached scsi generic sg2 type 0
...
```

In this case, it shows up as /dev/sdc (note sdc inside the square brackets above).

## 1) Check to see if the automounter has mounted the SD Card

### Note:

 There may be more than one partition (only one shown in the example below).

```
$ [df -h]
Filesystem Size Used Avail Use% Mounted on
...
/dev/sdc1 400M 94M 307M 24% /media/disk
...
```

### Note:

 the "Mounted on" field in the above and use that name in the umount commands below.

## 1. If so, unmount it

```
$ [umount /media/disk]
```

## 2. Start fdisk

Be sure to choose the whole device (/dev/sdc), not a single partition (/dev/sdc1).

```
$ [sudo fdisk /dev/sdc]
```

## 3. Print the partition record

So you know your starting point. Make sure to write down the number of bytes on the card (in this example, 2021654528).

```
Command (m for help): [p]

Disk /dev/sdc: 2021 MB, 2021654528 bytes
255 heads, 63 sectors/track, 245 cylinders
```

```
Units = cylinders of 16065 * 512 = 8225280 bytes

Device Boot Start End Blocks Id System
/dev/sdc1 * 1 246 1974240+ c W95 FAT32 (LBA)

Partition 1 has different physical/logical endings:
 phys=(244, 254, 63) logical=(245, 200, 19)
```

#### 4. Delete any partitions that are there already

```
Command (m for help): [d]
Selected partition 1
```

#### 5. Set the Geometry of the TF Card

If the print out above does not show 255 heads, 63 sectors/track, then do the following expert mode steps to redo the TF Card:

#### 6. Go into expert mode.

```
Command (m for help): [x]
```

#### 7. Set the number of heads to 255.

```
Expert Command (m for help): [h]
Number of heads (1-256, default xxx): [255]
```

#### 8. Set the number of sectors to 63.

```
Expert Command (m for help): [s]
Number of sectors (1-63, default xxx): [63]
```

#### 9. Now Calculate the number of Cylinders for your TF Card.

```
#cylinders = FLOOR (the number of Bytes on the TF Card (from above) / 255 / 63 / 512)

So for this example: 2021654528 / 255 / 63 / 512 = 245.79. So we use 245 (i.e. truncate, don't round).
```

#### 10. Set the number of cylinders to the number calculated.

```
Expert Command (m for help): [c]
Number of cylinders (1-256, default xxx): [enter the number you calculated]
```

## 11. Return to Normal mode.

```
Expert Command (m for help): [x]
```

## 12. Print the partition record to check your work

```
Command (m for help): [p]

Disk /dev/sdc: 2021 MB, 2021654528 bytes
255 heads, 63 sectors/track, 245 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

 Device Boot Start End Blocks Id System

```

## 13. Create the FAT32 partition for booting and transferring files from Windows

```
Command (m for help): [n]
Command action
 e extended
 p primary partition (1-4)
[p]
Partition number (1-4): [1]
First cylinder (1-245, default 1): [(press Enter)]
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-61, default 61): [+5]

Command (m for help): [t]
Selected partition 1
Hex code (type L to list codes): [c]
Changed system type of partition 1 to c (W95 FAT32 (LBA))
```

## 14. Mark it as bootable

```
Command (m for help): [a]
Partition number (1-4): [1]
```

## 15. Create the Linux partition for the root file system

```
Command (m for help): [n]
Command action
 e extended
 p primary partition (1-4)
[p]
Partition number (1-4): [2]
First cylinder (7-61, default 7): [(press Enter)]
Using default value 52
Last cylinder or +size or +sizeM or +sizeK (7-61, default 61): [(press Enter)]
Using default value 245
```

## 16. Print to Check Your Work

```
Command (m for help): [p]

Disk /dev/sdc: 2021 MB, 2021654528 bytes
255 heads, 63 sectors/track, 245 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

 Device Boot Start End Blocks Id System
/dev/sdc1 * 1 6 409626 c W95 FAT32 (LBA)
/dev/sdc2 7 61 1558305 83 Linux
```

## 17. Save the new partition records on the TF Card

This is an important step. All the work up to now has been temporary.

```
Command (m for help): [w]
The partition table has been altered!

Calling ioctl() to re-read partition table.

WARNING: Re-reading the partition table failed with error 16: Device or resource
busy.
The kernel still uses the old table.
The new table will be used at the next reboot.

WARNING: If you have created or modified any DOS 6.x
partitions, please see the fdisk manual page for additional
information.
Syncing disks.
```

## 18. Format the partitions

The two partitions are given the volume names LABEL1 and LABEL2 by these commands. You can substitute your own volume labels.

```
$ [sudo mkfs.msdos -F 32 /dev/sdc1 -n LABEL1]
mkfs.msdos 2.11 (12 Mar 2005)

$ [sudo mkfs.ext3 -L LABEL2 /dev/sdc2]
mke2fs 1.40-WIP (14-Nov-2006)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
195072 inodes, 389576 blocks
19478 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=402653184
12 block groups
32768 blocks per group, 32768 fragments per group
16256 inodes per group
Superblock backups stored on blocks:
 32768, 98304, 163840, 229376, 294912

Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information:
```

### Note:

📖 After formatting and dividing into FAT and EXT3 under the ubuntu system, the FAT needs reformatting under the windows system, otherwise, start-up with SD card will not be possible.

# Appendix 4: The Setup Of TFTP Server

## 1) Install client

```
$>sudo apt-get install tftp-hpa
$>sudo apt-get install tftpd-hpa
```

### 1. Install inet

```
$>sudo apt-get install xinetd
$>sudo apt-get install netkit-inetd
```

### 2. Configure the server

First, create tftpboot under the root directory, and set the properties as "a random user can write and read"

```
$>cd /
$>sudo mkdir tftpboot
$>sudo chmod 777 tftpboot
```

Secondly, add in /etc/inetd.conf:

```
$>sudo vi /etc/inetd.conf //copy the follow word to this file
```

```
tftpd dgram udp wait root /usr/sbin/in.tftpd /usr/sbin/in.tftpd -s /tftpboot
```

Then, reload inetd process:

```
$>sudo /etc/init.d/inetd reload
```

Finally, enter the directory /etc/xinetd.d/, and create a new file tftp and put the designated content into file tftp:

```
$>cd /etc/xinetd.d/
$>sudo touch tftp
$>sudo vi tftp //copy the follow word to tftp file
service tftp
{
```

```
disable = no
socket_type = dgram
protocol = udp
wait = yes
user = root
server = /usr/sbin/in.tftpd
server_args = -s /tftpboot -c
per_source = 11
cps = 100 2
}
```

### 3. Reboot the server:

```
$>sudo /etc/init.d/xinetd restart
$>sudo in.tftpd -l /tftpboot
```

### 4. Test the server

Conduct a test; create a file under folder /tftpboot

```
$>touch abc
```

Enter into another folder

```
$>tftp 192.168.1.15 (192.168.1.15was the server IP)
$>tftp> get abc
```

If the download is successful the server has been installed.

## Appendix 5: FAQ

Please access [http://www.elinux.org/DevKit8600\\_FAQ](http://www.elinux.org/DevKit8600_FAQ).

## Appendix 6: ESD Precautions & Handling Procedures

Please note that the board is supplied outside of an enclosure and some or all components are exposed. Therefore, extra attention must be paid to ESD (electrostatic discharge) precautions. To effectively prevent electrostatic damage, please follow the steps below:

- Avoid carpets in cool, dry areas. Leave development kits in their anti-static packaging until ready to be installed.
- Dissipate static electricity before handling any system components (development kits) by touching a grounded metal object.
- If possible, use antistatic devices, such as wrist straps and floor mats.
- Always hold an evaluation board by its edges. Avoid touching the contacts and components on the board.
- Take care when connecting or disconnecting cables. A damaged cable can cause a short in the electrical circuit.
- Prevent damage to the connectors by aligning connector pins before you connect the cable. Misaligned connector pins can cause damage to system components at power-on.
- When disconnecting a cable, always pull on the cable connector or strain-relief loop, not on the cable itself.



### Warning:

 This is a class A product. In a domestic environment this product may cause radio interference in which case the user may be required to take adequate measures.

# **Appendix 7: Technical support & Warranty**

Embest Technology Co., Ltd. established in March of 2000, is a global provider of embedded hardware and software. Embest aims to help customers reduce time to market with improved quality by providing the most effective total solutions for the embedded industry. In the rapidly growing market of high end embedded systems, Embest provides comprehensive services to specify, develop and produce products and help customers to implement innovative technology and product features. Progressing from prototyping to the final product within a short time frame and thus shortening the time to market, and to achieve the lowest production costs possible. Embest insists on a simple business model: to offer customers high-performance, low-cost products with the best quality and service.

## **7.1 Technical support service**

Embest provides one year of free technical support for all products. The technical support service covers:

- Embest embedded platform products software/hardware materials
- Assistance to customers with regards to compiling and running the source code we offer.
- Troubleshooting problems occurring on embedded software/hardware platforms if users have followed the instructions provided.
- Judge whether a product failure exists.

The situations listed below are not covered by our free technical support service, and Embest will handle the situation at our discretion:

- Customers encounter issues related to software or hardware during their development process

- Issues occur when users compile/run the embedded OS which has been modified by themselves.
- Customers encounter issues related to their own applications.
- Customers experience problems caused by unauthorised alteration of our software source code

## **7.2 Maintenance service clause**

1. Product warranty will commence on the day of sale and last 12 months provided the product is used under normal conditions
2. The following situations are not covered by the warranty, Embest will charge service fees as appropriate:
  - Customers fail to provide valid proof of purchase or the product identification tag is damaged, unreadable, altered or inconsistent with the product.
  - Products are subject to damage caused by operations inconsistent with their specification;
  - Products are subject to damage in either appearance or function due to natural disasters (flood, fire, earthquake, lightning strike or typhoon) or natural aging of components or other force majeure;
  - Products are subject to damage in appearance or function due to power failure, external forces, water, animals or foreign materials;
  - Products malfunction due to disassembly or alteration of components by customers, or repair by persons or organizations unauthorized by Embest Technology, or alteration from factory specifications, or configured or expanded with components that are not provided or recognized by Embest Technology;
  - Product failures due to the software or systems installed by customers, inappropriate software settings or computer viruses;
  - Products purchased from unauthorized merchants;

- Embest Technology takes no responsibility for fulfilling any warranty (verbal or written) that is not made by Embest Technology and not included in the scope of our warranty.
- 3. Within the period of warranty, the cost for sending products to Embest should be paid by the customer. The cost for returning the product to the customer will be paid by Embest. Any returns in either direction occurring after the warranty period has expired should be paid for by the customer.
- 4. Please contact technical support with any repair requests.

**Note:**

 Embest Technology will not take any responsibility for products returned without the prior permission of the company.

## **7.3 Basic guidelines for protection and maintenance of LCDs**

1. Do not use finger nails or other hard sharp objects to touch the surface of the LCD
2. Embest recommends purchasing specialist wipes to clean the LCD after long time use, avoid cleaning the surface with fingers or hands as this may leave fingerprints or smudges.
3. Do not clean the surface of the screen with unsuitable chemicals

**Note:**

 Embest do not supply a maintenance service for LCDs. We suggest the customer immediately checks the LCD once in receipt of the goods. In the event that the LCD does not run or shows no display, the customer should inform Embest within 7 business days of delivery.

## 7.4 Value Added Services

We will provide following value added services:

- Driver development based on Embest embedded platforms for devices such as: serial ports, USB interface devices, and LCD screens.
- Control system transplantation, BSP driver development, API software development.
- Other value added services including supply of power adapters and LCD parts.
- Other OEM/ODM services.
- Technical training.

Please contact Embest with any technical support queries:



<http://www.embest-tech.com/contact-us.html>